Oracle Utilities Application Framework

Version 4.5.0.1.3

Administrative Guide

Revision 12.0

G10560-01

June 2024

ORACLE®

Notices

Oracle Utilities Application Framework Version 4.5.0.1.3 Administrative Guide

G10560-01

Copyright Notice

Copyright © 2024, Oracle and/or its affiliates.

License Restrictions

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or de-compilation of this software, unless required by law for interoperability, is prohibited.

Warranty Disclaimer

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

Restricted Rights Notice

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

> U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

Hazardous Applications Notice

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for

use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Trademark Notice

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

Third-Party Content, Products, and Services Disclaimer

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Preface

About this Document

This document explains the general, security, user, designing, developing, and scripting options available in Oracle Utilities Application Framework (OUAF). It describes the user interface, database, configuration, and reporting tools available in OUAF. In addition, it provides information on how to configure incoming and outgoing messages and how to integrate Lightweight Directory Access Protocol (LDAP), Oracle Identity Manager (OIM), and Batch Scheduler with Oracle Revenue Management and Billing (ORMB).

Intended Audience

This document is intended for the following audience:

- System Administrators
- Consulting Team
- Implementation Team
- Development Team

Organization of the Document

The information in this document is organized into the following chapters:

Chapter Number	Chapter Name	Chapter Description
Chapter 1	General Options	Explains how to define installation options and different languages in the application. It explains how to define countries, currencies, time zones, geographic types, work calendars, display profiles, phone types, characteristic types, and foreign key references in the application. It also explains how to define feature and master configurations in the application.
Chapter 2	Security and User Options	Explains the different levels of security that you can define in the application. It explains the different types of encryption and masking supported in the application. It explains how to define application services, security types, user groups, access groups, data access roles, users, and cryptography keys in the application. It also describes how to implement object erasure to maintain the data privacy.
Chapter 3	User Interface Tools	Explains how to define menu options, error messages, zone types, zones, different types of portals, navigation keys, and navigation options in the application.
Chapter 4	Database Tools	Explains how to define tables, fields, maintenance objects, lookup fields, and extendable lookups in the application. It explains how to configure various features, such as audit trail, revision control, bundling, and information lifecycle management in the application.
Chapter 5	Configuration Tools	Describes the business object and how to define business objects in the application. It explains how to define business services, UI maps, process flows, and data areas in the application. It explains

Chapter Number	Chapter Name	Chapter Description
		the difference between algorithm types and algorithms. It explains how to define algorithms and algorithm types in the application. It describes the different types of scripts, such as BPA script, plugin script, service script, and groovy script. In addition, it explains how to define scripts and functions in the application.
Chapter 6	To Do Lists	Describes the To Do list and To Do entries. It explains how to define To Do types and To Do roles. It also explains the different ways in which a To Do can be created in the application.
Chapter 7	Background Processes	Explains the background processing and various concepts related to the background processing. It explains how to define batch controls and submit batch jobs in the application. It explains how to track the background process and how to assess the overall health of the application. It also describes how to configure various types of request in the application.
Chapter 8	Attachments	Explains the attachment functionality, supported content types, and various types of attachment modes used in the application.
Chapter 9	Monitoring and Analytics	Explains the various tools provided to support system monitoring including the tools related to monitoring and analytics, and reports.
Chapter 10	External Messages	Explains how to support incoming messages using the inbound web services and outgoing messages using the outbound messages. It explains how to create remote messages for the mobile application.
Chapter 11	Integrations	Explains how to integrate LDAP, Oracle Identity Manager, and Batch Scheduler with the application. It describes business flags and explains how to create business flags in the application.
Chapter 12	Market Transaction Management	Explains the various entities and aspects involved in the configuration and maintenance of the market transactions.
Chapter 13	Content Migration Assistant (CMA)	Provides an overview for the Content Migration Assistant (CMA) tool. It explains how to configure the CMA tool and use various processes in the tool while importing or exporting the data.
Chapter 14	Facts	Explains how to use the Facts maintenance object for configuring custom entities and supporting automated workflows for a variety of applications.
Chapter 15	Conversion	Explains the overall conversion process and the tools provided to support it.
Chapter 16	Data Export	Describes the concepts and tools related to various types of data export methods supported by the product.

Related Documents

You can refer to the following documents for more information:

Document Name	Description
Oracle Utilities Application Framework Version 4.5.0.1.3 Business Process Guide	Explains how to get acquainted with the user interface. It explains the different types of pages or portals that you may come across in the application. It explains how to set the user preferences and how to create, manage, assign, and complete a To Do in the application. It

Copyright © 2024, Oracle and/or its affiliates.

Document Name	Description
	also explains how to submit reports and view historic reports in the application.

Access to Oracle Support

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Copyright © 2024, Oracle and/or its affiliates.

Contents

Notices i

Preface: About this Document iii
- Intended Audience iii
- Organization of the Document iii
- Related Documents iv
- Access to Oracle Support v

Chapter 1: General Options 1
- Defining Installation Options 2
  - Installation Options - Main 2
  - Installation Options - Messages 3
  - Installation Options - Algorithms 3
  - Installation Options - Accessible Modules 5
  - Installation Options - Installed Products 5
- Support For Different Languages 6
  - User Language 6
  - Customer Language 7
  - Defining Languages 7
- Defining Countries 8
  - Country - Main 9
  - Country - States 9
- Defining Currency Codes 9
- Defining Time Zones 10
  - Designing Time Zones 10
  - Setting Up Time Zones 11
  - Setting Up Seasonal Time Shift 12
- Defining Geographic Types 12
- Defining Work Calendar 12
- Defining Display Profiles 13
  - Additional Hijri Date Configuration 15
- Defining Phone Types 15
- Defining Feature Configurations 15
  - Feature Configuration - Main 16
  - Feature Configuration - Messages 16
- Defining Master Configurations 16

Chapter 2: Security & User Options 19
- Application Security 20
  - Understanding Application Security 20
    - Users and User Groups 20
    - Secured Objects and Application Services 23
    - Field Level Security 26
    - Importing Security Configuration from an External Source 26
  - Defining Application Services 27
  - Defining Security Types 27

Copyright © 2024, Oracle and/or its affiliates.

Security Type - Main....27
Defining User Groups....28
Defining Users....29
Application Security Query....29
Row Security....29
Defining Access Groups....30
Defining Data Access Roles....30
Data Access Role - Main....30
Data Access Role - Access Group....31
Data Privacy....31
Encryption and Masking....31
System Encryption....32
User Interface Masking....32
Application Encryption....36
The Approach to Implementing Object Erasure....38
Configuring a Maintenance Object for Erasure....38
Manage Erasure Schedule Algorithm....38
Monitoring the Schedule and Performing Erasure....39
Erasing User Information By Obfuscation....39
Viewing an Object's Erasure Status....39
Viewing Erasure Configuration....40
Archiving the Object Erasure Schedule....40
Data Redaction on Export....40
Defining Redaction Rules....42
Obfuscation Configuration....42
Cryptography Keys....43
Understanding Key Rings....43
RSA Signature Keys....43
OAuth Keys....43
File Encryption and Signing Keys....44
Defining Key Rings....44

Chapter 3: User Interface Tools....47
Understanding Portals and Zones....48
Setting Up Portals and Zones....51
Defining Zone Types....51
Defining Zones....52
Overriding Parameter Values....52
Overriding Zone Parameters....52
Zone Parameter Details....53
Data Explorer Graph Configuration....84
Pagination Configuration....91
Use Data Explorer for Derived Data....92
Configuring Timeline Zones....93
Defining Context-Sensitive Zones....94
Defining Portals....94
Defining Menu Options....95
Defining Navigation Keys....96
Navigation Key Types....96
Navigation Key vs Navigation Option....96
The Flexibility of Navigation Keys....96
Linking to External Locations....97
Overriding Navigation Keys....97
Maintaining Navigation Keys....97
Defining Navigation Options....98

Copyright © 2024, Oracle and/or its affiliates.

Navigation Option - Main.....99
Navigation Option - Tree.....100
Understanding System Messages.....101
Defining System Messages.....101
Defining Message Categories.....101
Defining Messages.....101
Understanding Unified Search.....102
Trees.....104
Defining Trees.....106
Contextual Insights.....106
Insight Types.....107
Defining Insight Types.....109
Insight Groups.....109
Defining Insight Groups.....110
Color Configuration.....110
Defining Colors.....110
Color Options.....110
Defining Display Icons.....111
Custom Look and Feel Options.....111
User Interface.....111
UI Map Help.....112

Chapter 4: Database Tools.....113
Understanding The Data Model.....114
Fields.....114
Tables.....116
Maintenance Objects.....119
Defining Valid Values.....120
Characteristic Types.....121
Foreign Key References.....123
Defining Tables.....124
Defining Fields.....125
Defining Maintenance Objects.....125
Maintenance Object - Main.....126
Maintenance Object - Options.....126
Maintenance Object - Algorithms.....126
Maintenance Object - Maintenance Object Tree.....129
Defining Lookups.....129
Defining Extendable Lookups.....130
Extendable Lookup Advanced Topics.....130
Defining Characteristic Types.....131
Defining Foreign Key References.....132
Audit Trails.....133
Captured Information.....133
How Auditing Works.....133
The Audit Trail File.....134
How To Enable Auditing.....134
Turn On Auditing For a Table.....134
Specify The Fields and Actions To Be Audited.....134
Audit Queries.....135
Audit Query by User.....135
Audit Query by Table / Field / Key.....136
Bundling.....136
About Bundling.....136
Sequencing of Objects in a Bundle.....137

Copyright © 2024, Oracle and/or its affiliates.

Recursive Key References....137
Owner Flags on Bundled Entities....137
Configuring Maintenance Objects for Bundling....138
Working with Bundles....139
Creating Export Bundles....139
Creating and Applying Import Bundles....140
Editing Export Bundles....141
Editing Import Bundles....141
Revision Control....141
About Revision Control....141
Turning On Revision Control....141
Configuring Maintenance Objects for Revision Control....142
Working with the Revision Control Zones....142
Checking Out an Object....143
Checking In an Object....143
Reverting Changes....143
Forcing a Check In or Restore....143
Deleting an Object....143
Restoring an Object....144
Working with the Revision Control Portal....144
Revision Control Search....144
Information Lifecycle Management....145
The Approach to Implementing Information Lifecycle Management....145
Batch Processes....146
Eligibility Algorithm....147
Enabling ILM for Supported Maintenance Objects....147
Ongoing ILM Tasks....148
Archived Foreign Keys....149

Chapter 5: Configuration Tools....151
Business Objects....152
Understanding Business Objects....152
What Is A Business Object?....152
Business Object Inheritance....159
Each Business Object Can Have A Different Lifecycle....160
BO Algorithm Execution Summary....167
Granting Access To Business Objects....168
Defining Business Objects....168
Advanced BO Tips and Techniques....171
Creating a Physical Business Object....171
Managing To Do Entries....171
Submitting a Batch Job....172
Defining Status Reasons....172
Data Areas....172
Defining Data Areas....173
Data Area - Main....173
Data Area - Schema....173
Business Services....174
Service Program....174
Defining Business Services....175
Business Service - Main....175
Business Service - Schema....175
Useful Services and Business Services....176
User Interface (UI) Maps....179
Defining UI Maps....181

Copyright © 2024, Oracle and/or its affiliates.

UI Map - Main.... 181
UI Map - Schema.... 182
UI Map Attributes and Functions.... 182
UI Map Standards.... 238
Ensuring Unique Element IDs for UI Maps.... 251
Process Flows.... 252
Understanding Process Flows.... 252
Designing Process Flows.... 254
Defining Process Flow Types.... 256
Action Method.... 256
Defining Action Methods.... 258
Algorithms.... 258
The Big Picture Of Algorithms.... 258
How To Add A New Algorithm.... 260
Minimizing The Impact Of Future Upgrades.... 260
Defining Algorithm Types.... 260
Defining Algorithms.... 261
Advanced Algorithm Tips.... 262
Script Options.... 262
The Big Picture Of Scripts.... 262
Scripts Are Business Process-Oriented.... 262
A Script Is Composed Of Steps.... 263
A Script May Declare Data Areas.... 263
Securing Script Execution.... 263
The Big Picture Of BPA Scripts.... 263
How To Invoke Scripts.... 264
Developing and Debugging Your BPA Scripts.... 264
Launching A Script From A Menu.... 265
Launching A Script When Starting The System.... 265
Executing A Script When A To Do Entry Is Selected.... 265
The Big Picture Of Script Eligibility Rules.... 265
The Big Picture Of Server-Based Scripts.... 269
Additional Coding Options For Server Scripts.... 270
Plug-In Scripts.... 271
Service Scripts.... 272
Groovy Library Scripts.... 273
Debugging Server-Based Scripts.... 273
Maintaining Scripts.... 273
Script - Main.... 273
Script - Step.... 275
Script - Data Area.... 317
Script - Schema.... 318
Script - Eligibility.... 318
Merging Scripts.... 319
Script Merge.... 320
Maintaining Functions.... 323
Function - Main.... 323
Function - Send Fields.... 324
Function - Receive Fields.... 324
Maintaining Managed Content.... 325
Defining Entity Tags.... 325
Advanced Schema Topics.... 326
Schema Nodes and Attributes.... 326
UI Hint Syntax.... 347
Schema Designer.... 356
Schema Viewer.... 357

Copyright © 2024, Oracle and/or its affiliates.

Business Event Log....357
Miscellaneous Topics....358
Javadocs Viewer....358
Module Configuration....358
Menu Item Suppression....359
Menu Suppression....359
Turn Off A Function Module....359
Global Context Overview....359
System Data Naming Convention....360
Base Product System Data....360
Implementation System Data....360
Accessibility Considerations....361
Referencing URIs....361
Validation Against an Allowlist....361
URI Substitution....361
External File Storage....362
Caching Overview....363
Expression Parser....365
Debug Mode....368
Prepare Issue Details....369
System Override Date....369
Additional Search Options....369
Extensions Dashboard....370

Chapter 6: To Do Lists....371
The Big Picture of To Do Lists....372
To Do Entries Reference A To Do Type....372
To Do Entries Reference A Role....372
The Lifecycle Of A To Do Entry....373
How Are To Do Entries Created?....374
Suppress the Creation of the To Do Entry....374
To Do Entries Created By Background Processes....375
To Do Entries Created By Algorithms....376
To Do Entries Created Manually....377
Refining a To Do Entry's Details at Creation Time....377
The Priority Of A To Do Entry....379
What Happens When You Work A To Do Entry....380
Monitoring A To Do Entry....380
To Do Entries Have Logs....380
Implementing Additional To Do Entry Business Rules....381
To Do Entries May Be Routed Out Of The System....381
To Do Information May Be Formatted By An Algorithm....381
Periodically Purging To Do Entries....381
Setting Up To Do Options....382
Installation Options....382
Messages....382
Feature Configuration....382
Defining To Do Roles....383
To Do Role - Main....383
To Do Role - To Do Types....383
Defining To Do Types....384
To Do Type - Main....384
To Do Type - Roles....384
To Do Type - Sort Keys....385
To Do Type - Drill Keys....385

Copyright © 2024, Oracle and/or its affiliates.

To Do Type - Message Overrides....386
To Do Type - To Do Characteristics....386
To Do Type - Algorithms....386
Implementing The To Do Entries....387

Chapter 7: Background Processes....389
Understanding Background Processes....390
Background Processing Overview....390
Parallel Background Processes....391
Optimal Thread Count....392
Parameters Supplied To Background Processes....392
Indicating a File Path or File Name....395
Encrypting and Signing Files....395
Manifest File....396
Sharing Customer Exported Data....397
Processing Errors....398
Error Post-Processing Logic....398
Post-Processing Logic....398
Timed Batch Processes....399
Monitor Background Processes....399
Plug-in Driven Background Processes....400
Ad-hoc Processes....401
Extract Record Processing....403
Uploading Records....412
How to Re-extract Information....413
How to Submit Batch Jobs....414
How to Track Batch Jobs....414
How to Restart Failed Jobs and Processes....414
Assessing Level of Service....414
Batch Resource Groups....414
Defining Batch Controls....415
Batch Control - Algorithms....417
Defining File Integration Types....418
On-Line Batch Submission....419
Batch Submission Creates a Batch Run....419
Jobs Submitted in the Background....419
Email Notification....419
Running Multi-Threaded Processes....420
Batch Jobs May End in Error....420
Submitting Jobs in the Future....420
Lifecycle of a Batch Job Submission....420
Granting Access To Batch Submission....421
Batch Job Submission Portal....421
Tracking Batch Processes....423
Batch Run Tree....424
Batch Run Tree - Main....424
Batch Run Tree - Run Control....425
Batch Day Tracking....425
Batch Day Dashboard....425
Batch Run Threads....427
The Big Picture of Requests....427
Request Type Defines Parameters....427
Previewing and Submitting a Request....427
To Do Summary Email....428
Defining a New Request....428

Copyright © 2024, Oracle and/or its affiliates.

Setting Up Request Types.... 428
Maintaining Requests....429

Chapter 8: Attachments....431
Attachment Overview.... 432
Configuring Your System for Attachments....433
Maintaining Attachments....434
Adding Attachments.... 434

Chapter 9: Monitoring and Analytics....435
Reporting Tool Integration.... 436
The Big Picture Of Reports....436
Integration with BI Publisher.... 436
How To Request Reports....437
Viewing Reports....437
Configuring The System To Enable Reports.... 437
Configuring BI Publisher Reports....437
Defining Reporting Options.... 438
Defining Report Definitions.... 438
Sample Reports Supplied with the Product.... 439
How to Use a Sample Report Provided with the System.... 439
How To Define A New Report....440
Use a Sample Report as a Starting Point....440
Publishing Reports in BI Publisher....441
Designing Your Report Definition.... 441
Service Health Check.... 443
Health Check Configuration....444
Viewing Health Check Results....444
Introducing Health Check Conditions....445
Measuring Performance.... 445
Understanding Performance Targets....445
Performance Target Objects Overview....445
Calculating and Displaying Performance Targets.... 446
Performance Target Metrics and Metric Types.... 446
Performance Target Categories and Types.... 446
Performance Targets Define Specific Metrics....447
Objects Linked to a Performance Target....447
Creating Performance Target Zones....447
Setting Up Performance Target Configuration....448
Performance Target Category Lookup.... 448
Defining Performance Target Types....448
Maintaining Performance Targets....448
Capturing Statistics.... 448
Understanding Statistics....448
Configuring Your System for Statistics.... 449
Defining and Monitoring Statistics.... 449
Creating Cube Views....450
Understanding Cube Viewer....450
Cube Viewer Components.... 450
Cube Configuration Components.... 454
Configuring Cube Types....454
Maintaining Cube Types....454
Cube Type Advanced Topics....455
Product Use Metrics.... 455

Copyright © 2024, Oracle and/or its affiliates.

Chapter 10: External Messages..... 457
Incoming Messages.....458
Inbound Web Services.....458
Understanding Inbound Web Services..... 458
Configuring Inbound Web Service Options.....463
Deploying Inbound SOAP Web Services.....472
Guaranteed Delivery..... 474
Outgoing Messages.....474
Outbound Messages.....474
Polling Outbound Messages Using OSB..... 475
Batch Message Processing.....475
Real Time Messages.....476
Designing the System for Outbound Messages..... 476
Configuring the System for Outbound Messages..... 479
Managing Outbound Messages.....488
Web Service Adapters.....489
Understanding Web Service Adapters.....489
Setting Up Web Service Adapters.....491
Sending Email.....491
Web Service Category..... 492
Defining Web Service Categories..... 492
JMS Message Browser..... 492
Oracle Integration Cloud Catalog.....493
Web Service Catalog Configuration.....493
Web Service Catalog Master Configuration.....494
Maintaining the Web Service Catalog.....494
XAI Documentation Note.....494

Chapter 11: Integrations.....495
Creating Application Links in External Applications.....496
LDAP Integration.....497
LDAP Integration Overview..... 497
Configuring LDAP Integration.....498
LDAP Mapping.....499
Oracle Identity Manager Integration..... 502
Provision Users with SCIM Open Standard..... 503
User Redirection..... 504
Batch Scheduler Integration..... 504
Data Synchronization.....505
Understanding Data Synchronization..... 505
Accumulative Data Synchronization..... 505
Generic Data Synchronization.....507
Maintaining Sync Requests..... 508
Operational Analytics..... 509
Understanding Operational Analytics.....509
Calendar and Time Dimensions..... 510
Maintaining Bucket Configurations..... 511
Defining Analytics Options..... 511
Defining Analytics Tables..... 512
Defining Analytics Dimensions.....512
Defining Analytics Facts..... 512
Defining Analytics Attribute Mapping.....513
Analytics Integration.....513

Copyright © 2024, Oracle and/or its affiliates.

Understanding Analytics Integration....513
Business Flags....514
Understanding Business Flags....514
Standard Name....514
Business Flag Type Defines Behavior for a Standard Name....515
Business Flag Type Algorithms....515
Objects Linked to a Business Flag....515
Impacted Business Process....515
Dates....516
Creating Business Flags....516
Confidence....517
Setting Up Business Flag Configuration....517
Standard Name Category Characteristic Type....518
Standard Name Lookup....518
Business Process Lookup....518
Integration Configuration....518
Defining Business Flag Types....518
Maintaining Business Flags....518

Chapter 12: Market Transaction Management....521
Understanding Market Transaction Management....522
Configuring Market Transaction Management....522
Defining Market Configurations....522
Defining Market Message Types....522
Defining Market Process Types....523
Maintaining Market Transactions....523
Maintaining Market Processes....523
Maintaining Market Process Events....523
Maintaining Market Messages....524

Chapter 13: Content Migration Assistant (CMA)....525
Understanding CMA....526
Types of Migrations....527
Migrating Configuration Data....528
Migrating Business Data....528
Data with System Generated Primary Keys....531
Importing Data that References a User....532
Additional Considerations....532
Versions and CMA....533
CMA Configuration....534
Master Configuration - Migration Assistant....534
Migration Plans....535
Defining a Migration Plan....535
Understanding the BO Filtering Process....536
Migration Plans for Objects with XML-Embedded Links....537
Defining a Migration Request....538
Identifying Tables to Exclude From Migrations....539
Configuring Custom Objects for Migration....539
The CMA Execution Process....540
Exporting a Migration....541
Migration Data Set Export....542
Export Lifecycle....542
Importing and Applying a Migration....543
Import Step....544

Copyright © 2024, Oracle and/or its affiliates.

Compare Step....545
Approval Step....548
Apply Step....549
Adjusting Data Prior to Comparing....553
Import Process Summary....554
Cancelling a Data Set....561
Additional Note Regarding Imports....561
Caching Considerations....562
Maintaining Import Data....562
Running Batch Jobs....563
CMA Reference....565
Framework-Provided Migration Configuration....565

Chapter 14: Facts....569
Fact Is A Generic Entity....570
Fact's Business Object Controls Everything....570
Fact Supports A Log....570

Chapter 15: Conversion....571
Understanding The Conversion Process....572
Conversion Entities....573
Conversion Steps....573
Load Legacy Data Into Staging Tables....573
Validate Information In The Staging Tables....574
Allocate Production Keys....575
XML Resolution....577
Insert Production Data....578
Validate Production....578
A Note About Keys....579
Multiple Owners In A Single Database....579
Conversion Entity Dashboard....580

Chapter 16: Data Export....583
Understanding Data Export....584
Generalized Data Export....584
Specialized Data Export....589
Defining Data Export Control....590
Generalized Export Dashboard....590

Copyright © 2024, Oracle and/or its affiliates.

Chapter 1

General Options

Topics:

- Defining Installation Options
- Support For Different Languages
- Defining Countries
- Defining Currency Codes
- Defining Time Zones
- Defining Geographic Types
- Defining Work Calendar
- Defining Display Profiles
- Defining Phone Types
- Defining Feature Configurations
- Defining Master Configurations

This section describes control tables that are used throughout your product.

Defining Installation Options

The topics in this section describe the various installation options that control various aspects of the system.

Installation Options - Main

Select **Admin** > **General** > **Installation Options - Framework** to define system wide installation options.

Description of Page

The **Environment ID** is a unique universal identifier of this instance of the system. When the system is installed, the environment id is populated with a six digit random number. While it is highly unlikely that multiple installs of the system at a given implementation would have the same environment ID, it is the obligation of the implementers to ensure that the environment ID is unique across all installed product environments.

System Owner will be **Customer Modification**.

The **Admin Menu Order** controls how the various control tables are grouped on Admin.

- If you choose **Functional**, each control table appears under a menu item that corresponds with its functional area. Note, the menu that is used when this option is chosen is the one identified with a menu type of **Admin**.
- If you choose **Alphabetical**, each control table appears under a menu item that corresponds with its first letter, using a Roman alphabet. For example, the Language control table will appear under the L menu item entry.

Note: The **Alphabetical** option only supports the Roman alphabet. For languages that do not use the Roman alphabet, the recommendation is to configure the system for the **Functional** setting.

CAUTION: In order to improve response times, installation options are cached the first time they are used after a web server is started. If you change the Admin Menu Order and you don't want to wait for the cache to rebuild, you must clear the cached information so it will be immediately rebuilt using current information. Refer to Caching Overview for information on how to clear the system login cache (this is the cache in which installation options are stored).

The **Language** should be set to the primary language used by the installation. Note that if multiple languages are supported, each user may define their preferred language.

The **Currency Code** is the default currency code for transactions in the product.

If your product supports effective dated characteristics on any of its objects, define the date to be used as the **Characteristic Default Date** on objects without an implicit start date. The date you enter in this field will default when new characteristics are added to these objects (and the default date can be overridden by the user).

Active Owner displays the owner of newly added system data (system data is data like algorithm types, zones, To Do types, etc.). This will be **Customer Modification** unless you are working within a development region.

Country and **Time Zone** represent the default country and time zone that should be used throughout the application.

CAUTION: In most implementations, the time zone defined here matches the database time zone. However, if there is some reason that the database time zone does not match the installation time zone, an implementation may configure a setting in the properties file to automatically convert data from the database time zone to the time zone defined here when displaying dates. Note that when this property setting is defined, changes to the installation time zone will require the server and the thread pool workers to be restarted in order for the changes to take effect.

Turn on **Seasonal Time Shift** if your company requires seasonal time shift information to be defined. Note that this is currently only applicable to Oracle Customer Care and Billing > Interval Billing functionality.

Copyright © 2024, Oracle and/or its affiliates.

Installation Options - Messages

Select **Admin** > **General** > **Installation Options - Framework** and the **Messages** tab to review or enter messages that will appear throughout the application when a given event occurs.

The **Message** collection contains messages that are used in various parts of the system. For each message, define the **Installation Message Type** and **Installation Message Text**. The following table describes the **Message Types** provided by the framework product and how they are used in the system. Your specific product may have introduced addition message types.

Message Type	How The Message Is Used
Company Title for Reports	This message appears as a title line on the sample reports provided with the system. Generally it is your company name. It is only used if you have installed reporting functionality and are using the sample reports (or have designed your reports to use this message).
Domain Name	The text entered for this message type is meant to identify the environment and can be used for logic where it is useful to indicate the environment for a user. For example, the email sent when a background process finishes includes this information, if populated.

Installation Options - Algorithms

Select **Admin** > **General** > **Installation Options - Framework** and the **Algorithms** tab to review or enter the algorithms that should be evoked when a given event occurs.

The grid contains **Algorithms** that control important functions in the system. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence Number** and **Algorithm** for each system event. You can set the **Sequence Number** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

CAUTION: These algorithms are typically significant processes. The absence of an algorithm might prevent the system from operating correctly.

The following table describes each **System Event**.

System Event	Optional / Required	Description
Validate Email Attachment	Optional	Algorithms of this type are used to validate the attachments for size and total count while sending attachments using the Email service. Refer to Sending Email for more information
Address Geocoding	Optional	Algorithms of this type use Oracle Locator to retrieve latitude and longitude coordinates using address information.
Attachment Malware Scan	Optional	Algorithms of this type are invoked when adding attachment content. The algorithm receives the attachment ID

Copyright © 2024, Oracle and/or its affiliates.

System Event	Optional / Required	Description
		and is expected to use a malware scanning tool to verify the contents of the attachment. Only one algorithm is allowed. Refer to Enabling Malware Scan for more information.
Global Context	Optional	Algorithms of this type are called whenever the value of one of the global context fields is changed. Algorithms of this type are responsible for populating other global context values based on the new value of the field that was changed. Refer to Global Context Overview for more information.
Guaranteed Delivery	Optional	Algorithms of this type may be called by processes that receive incoming messages that should 'guarantee delivery'. Refer to Guaranteed Delivery for more information. The business service **F1-GuaranteedDelivery** may be used to invoke this plug-in spot.
Ldap Import	Optional	Algorithms of this type are called for operations on users, groups, and group memberships after they have been processed.
Ldap Import Preprocess	Optional	Algorithms of this type are called to preprocess data retrieved from LDAP.
Next To Do Assignment	Optional	This type of algorithm is used to find the next To Do entry a user should work on. It is called from the Current To Do zone when the user asks for the next assignment.
Reporting Tool	Optional	If your installation has integrated with a third party reporting tool, you may wish to allow your users to submit reports on-line using report submission or to review report history online. This algorithm is used by the two on-line reporting pages to properly invoke the reporting tool from within the system.
SMS Receive	Optional	This type of algorithm is used to provide SMS receive service. Only

Copyright © 2024, Oracle and/or its affiliates.

System Event	Optional / Required	Description
		one algorithm of this type should be plugged in.
SMS Send	Optional	This type of algorithm is used to provide SMS send service. If your installation uses the base algorithm that uses BPEL, you will need to create a feature configuration with the SMS Send Configuration feature type to define your Oracle BPEL server and service call details. If your installation has integrated with a third-party SMS service, you may want to override this algorithm type with your own implementation. Only one algorithm of this type should be plugged in.
To Do Information	Optional	We use the term To Do information to describe the basic information that appears throughout the system to describe a To Do entry. Plug an algorithm into this spot to override the system default "To Do information".
To Do Pre-creation	Optional	These types of algorithms are called when a To Do entry is being added. Refer to Refining a To Do Entry's Details at Creation Time for more information.

Installation Options - Accessible Modules

Select **Admin** > **General** > **Installation Options - Framework** and the **Accessible Modules** tab to view the list of accessible modules.

Description of Page

This page displays the full list of the application's function modules. A **Turned Off** indication appears adjacent to a module that is not accessible based on your system's module configuration setup.

Fastpath: Refer to Module Configuration for more information on function modules and how to turn off modules that are not applicable to your organization.

Installation Options - Installed Products

Select **Admin** > **General** > **Installation Options - Framework** and the **Installed Products** tab to view a read only summary of the products that are installed in the application version that you are logged into.

Description of Page

Copyright © 2024, Oracle and/or its affiliates.

The **Product Name** indicates the name of the "products" that are installed. The collection should include **Framework**, an entry for your specific product and an entry for **Customer Release**.

Release ID shows the current release of the application that is installed. This field is used by the system to ensure that the software that executes on your application server is consistent with the release level of the database.

The **Release ID Suffix**, **Build Number** and **Patch Number** further describe the details of your specific product release.

Note: If you have developed implementation-specific transactions, you can populate the release ID, release ID suffix, build number and patch number for the **Customer Release** entry to define the latest version of implementation-specific logic that has been applied to this environment. The framework provides a script (**F1CustRelM**) to allow these fields to be updated manually. In addition, the **F1-InstallationOptions** inbound web service provides operations to view and update those fields via a REST call. Alternatively, your implementation team may populate these fields as part of their upgrade scripts.

The **Product Status** is used to indicate if a product installed by default in the version you are logged into is **Active** or **Inactive**.

The **Display** column indicates the product whose name and release information should be displayed in the title bar. Only one product sets this value to **Yes**.

Owner indicates if this entry is owned by the base package or by your implementation (**Customer Modification**).

Product Type indicates if the product is a Parallel Application. A parallel application is one that is independent of, and does not conflict with, other parallel applications. Multiple parallel applications can be installed in the same database and application server.

Note: About Information. The information on this tab is used to populate the information displayed in the About window for your product.

Support For Different Languages

User Language

The language English (with the locale "en") is provided by the system and all system metadata are delivered with an English description or label. The system provides support for defining other languages and supports multiple languages in a single environment. System users can use the system in their preferred language, as long as a translation into that language has been provided. A user sees the system in the language defined on their user record. If enabled, users can use the Switch Language zone to switch to another supported language real time.

Supporting a New Language

Normally, setting up the system for another language (besides the delivered English language entry) is an implementation issue, not an administrative setup issue. However, there are several online administrative features that are used to set up a new language, and these are described here.

1. **Define a language code and indicate that it is enabled.** For details on this procedure, see Defining Languages.
2. **Install or confirm the existence of a language pack**. For an on-premise installation, if the product supplies a language pack with translations for the system metadata descriptions, follow the instructions for downloading the language pack and installing it into your environment. For a cloud installation, if the product supplies a language pack for a given language, no additional step is needed here. The language pack is already installed.
3. **Copy descriptions of all language-enabled tables from an existing translation (e.g., English).** The copied values act as placeholders while the strings are translated into the new language. It is necessary to do this as a first step in order to create records using the new language code created in the previous step. Language-based descriptions can be copied using a supplied batch process, **F1-LANG**. The batch copies all English labels in the system.

Copyright © 2024, Oracle and/or its affiliates.

The batch process will also update the new language rows with the translated metadata descriptions from the language pack, if installed.

4. **Translate additional content.** Translatable descriptions and labels for implementation data may be updated / entered in the application. First the user record must be updated to reference the new language. This may be done in one of the following ways:

   a. Switch to the new language using the Switch Language zone.
   b. If that zone is not available, navigate to the user page, assign the new language code to your User ID, sign out, and sign back in again.

   Any online functions that you access will use your new language code. You can change the language code for all users who plan to use/modify the new language.

Note: The language pack updates all language entries for base owned system data. If your implementation updates base owned labels and descriptions after applying a language pack, they will be overwritten the next time an updated language pack is applied. Note that most user facing labels and messages support defining an Override Label or Override Description. This information is not updated by the base product and should be utilized if your implementation desires a specific label or description.

Managing Multiple Languages

Once a non-English language has been set up, there are several points to note when adding or updating records that include translatable strings.

- When a new record is added through the user interface or through a REST service call or any other means that goes through the standard maintenance service, the system automatically generates entries for each supported language and copies the text of the entered description to all the generated languages.
- When the translatable string for a record is updated, only the user's language row is updated. The system does not ripple the change to other supported language entries (because it could be overriding a valid translation).
- When your system is upgraded to a new version, the new version typically includes new system data. The upgrade only includes the English language rows. Implementers should run the **F1-LANG** batch program to generate language rows for the your supported languages for this new data. If a new translation pack for your supported language is available at the time of upgrade, then running this process also applies the translated strings. (As mentioned above, for on-premise installations, you need to download and install the language pack before this step). If the language pack is not available at the time of upgrade, then the batch program should be run again when the language pack has been installed.
- New translatable objects may also be included in a patch / hot fix. In this situation, the **F1-LANG** batch program should be run after installing the patch in order to ensure there are language rows for your supported languages. Updated language packs are only provided for full release upgrades and not for hot fixes. So implementations may need to temporarily supply translations for the new strings until you update to the next release with the next updated translation pack.

Customer Language

Your specific product may also support capturing the language of a customer. Such that correspondence sent from the product may be produced in a language set on a customer record. Refer to your specific product’s documentation for more information about additional language support.

Defining Languages

Your product may support multiple languages. For example, the field labels, input text, and even outputs and reports can be configured to appear in a localized language. A language code for every potential language exists in the system to supply this information in various languages.

Select **Admin** > **General** > **Language** to define a language.

Copyright © 2024, Oracle and/or its affiliates.

This is a standard All-in-One portal.

When **Language Enable** is checked, the system adds a row for this language whenever a row is added in another language. For example, if you add a new currency code, the system will create language specific record for each language that has been enabled. You would only enable multiple languages if you have users who work in multiple languages. Languages that are configured as enabled, appear in the Switch Language zone.

> **Note:** For on premise applications, the login page displays all the languages that are enabled, allowing the user to toggle the login instructions in that language. The list of enabled languages is captured on the server at startup time. If a new language is enabled, contact your server administrator to refresh the server in order to see the new language displayed in the login page.

The **Locale** is a string containing three portions:

- ISO language code (lower case, required)
- ISO country code (upper case, optional)
- Variant (optional).

Hyphens separate the various portions, and multiple variants are possible.

The locale along with the collator strength control how the contents of grids and search results are sorted by the Java virtual machine (JVM) on your web server: The specific JVM in use by your particular hardware/OS configuration constrains the available **Locales**. Validating the **Locale** against the JVM is outside the scope of this transaction. This means you are responsible for choosing valid **Locales**.

Additionally, external systems that communicate with the product and provide a language use a format that matches the locale. For example in the Accept-Language header for REST web service calls, a requested language is provided.

The following are examples of valid locales:

Locale	Comments
en-US	American English
en-AU	Australian English
pt-BR	Brazilian Portuguese
fr-FR-EURO	European French
ja-JP	Japanese

The **Collator Strength** parameter is used for the Java collation API . This parameter controls whether, for example, upper and lower-case characters are considered equivalent, or how accented characters are sorted. Valid values for collator strength are **PRIMARY**, **SECONDARY**, **TERTIARY**, and **IDENTICAL**. If you leave this field blank, Java will use its default value for the language. We'd like to stress that the impact of each value depends on the language.

Please see https://docs.oracle.com/javase/7/docs/api/java/text/Collator.html for more information about the collator strength for your language.

Display Order indicates if this language is written **Left to Right** or **Right to Left**.

Note that all administrative control tables and system metadata that contain language-specific columns (e.g., a description) reference a language code. In addition, other tables may reference the language as a specific column. For example, on the User record you indicate the preferred language of the user.

Defining Countries

The topics in this section describe how to maintain countries.

Copyright © 2024, Oracle and/or its affiliates.

Country - Main

To add or review Country definitions choose **Admin** > **General** > **Country.**

The **Main** page is used to customize the fields and field descriptions that will be displayed everywhere addresses are used in the system. This ensures that the all addresses conform to the customary address format and conventions of the particular country you have defined.

Description of Page

Enter a unique **Country** and **Description** for the country.

The address fields that appear in the **Main** page are localization options that are used to customize address formats so that they conform to address requirements around the world. By indicating that an address field is **Optional**, you make that field available everywhere addresses for this country are used in the system. You can enter your own descriptions for the labels. These labels will appear wherever addresses are maintained in the system.

Note: Your specific product may also add the ability to mark an address field as **Required**. If that is available, then the product is also supplying appropriate validation in all places where a user defines an address.

Note: For any country where the **State** is enabled, the valid states for the country must be entered on the **Country - State** tab. When entering address constituents on a record that captures this detail, the value for State is verified against the data in the State table. For any country where there is a component of the address that represents a "state" but your implementation does not want to populate the valid states for that country, choose a different field such as County for this constituent (and define an appropriate label). When entering address constituents on a record that captures this detail, no validation is done for the County column.

Country - States

To maintain the states located in a country, choose **Admin** > **Country** > **Search** and navigate to the **State** page.

Description of Page

For any country where you have enabled the State switch, use the **State** collection to define the valid states in the **Country**.

- Enter the standard postal abbreviation for the **State** or province.
- Enter a **Description** for this state or province.

Defining Currency Codes

The currency page allows you to define display options related to currency codes that are used by your system. Use **Admin** > **General** > **Currency** to define the currency codes in which financial information is denominated.

This is a standard All-in-One portal.

Most attributes are self explanatory.

Note: Please contact your specific product to verify whether it supports a currency with more than 2 **Decimals**.

The **External Reference** may be used to define a currency whose currency code is more than three characters. To support that type of currency, define a currency code that uniquely identifies the currency in the system and use the external reference to define the currency code recognized by external products. Note that your specific integration with an external system where financial information is included may need to be adjusted to use the external reference instead of the system's currency code. Refer to the specific documentation for your integrations to confirm.

Copyright © 2024, Oracle and/or its affiliates.

Defining Time Zones

The following topics describe how to design and set up time zones.

Designing Time Zones

The Time Zone entity is used to define all the time zones where your customers may operate. Each time zone should define an appropriate Time Zone Name. This is a reference to an external source that defines time zones, their relationship to Greenwich Mean Time, whether the time zone follows any shifting for summer / winter time (daylight savings time) and when this shift occurs.

The following sections describe concepts and configuration topics related to managing time zones.

Note: Oracle Utilities Customer Care and Billing - Interval Billing applications customers should consult the topic *Time Issues* (search the Help index for "time issues") for specific information relating to that product's interval billing time related functionality.

The Base Time Zone

When designing your time zones, the first thing to determine is the base time zone. You may choose the time zone where the company's main office resides. Once this is done you can link the time zone code to the installation option as the base time zone. Refer to Installation Options - Main for more information.

Note: An attribute in the system properties file may be configured to indicate that the DB session time zone should be synchronized with the value defined on the Installation Options. Refer to the *Server Administration Guide* for more information.

If your company does business beyond your main office's time zone, define the other time zones where you may have customers or other systems with which you exchange data. At this point, your specific product may include configuration tables to capture default time zones, for example based on a postal code or geographic location.

Standard vs. Legal Time

The term **Legal Time** refers to the actual clock time as may be impacted by daylight saving (also called 'local time'). This time line has a gap hour on entry to the daylight saving period and a duplicate hour on exit from it.

The term **Standard Time** refers to a time line without any daylight saving shifts applied to it. This virtual time line has no gap hour on entry to the daylight saving period nor a duplicate hour on exit, i.e. it is continuous and non-ambiguous. As a general rule, it is recommended that all time sensitive data is stored in the standard time of the base time zone as defined on the installation options. This will prevent any confusion when analyzing data and will ensure that your algorithms do not have to perform any time zone or daylight saving shifting of data that may be stored in different time zones.

Depending on your specific product, entities may have their date/time information stored in either of the following options:

- Standard time of the base time zone (also called 'physical time').
- Standard time of another time zone related to the entity’s geographical location (also called 'logical time').
- Legal time of the base time zone.

The metadata definition of a table specifies whether any of its date/time fields are stored in standard time or not. While typically all date/time fields of a table that supports standard time would be stored in the same way the system allows each field to specify its own option. The metadata definition of such field specifies whether it is base time zone standard time (**Physical Standard Time**), standard time of some other time zone associated with the entity (**Legal Standard Time**) or stored in base time zone legal time (**Referenced Time Zone**).

Copyright © 2024, Oracle and/or its affiliates.

Storage vs. Display

Regardless of the time zone option used for storing the data in standard time, all date/time information is entered and displayed on the user interface in the legal time of its respective time zone. If a field is defined to be stored in the standard time of the base or other time zone, the user interface is designed to accept the data in the legal time of the respective time zone and shift it to the standard time of the same time zone before saving the data and performs the reverse shift from standard time back to legal time before displaying the data to the user.

It is important to understand that time zone and daylight saving time conversions to and from standard time are performed behind the scenes as part of the user interface layer. Once data reaches the server it is already in its storage time option. The user should not be aware of these time conversions.

Time shifting between standard and legal times does not happen automatically. When dealing with fields stored in standard time, their data entry and display need to explicitly include time shifting features as part of their user interface and zone configuration.

Date / Time Schema Elements

When defining date / time fields in a business object schema, schema attributes can be used to define whether or not data should be stored in standard time for the base time zone or if it should be stored in the standard time of another time zone (related to the data).

By default data stored in standard time is displayed in the corresponding time zone's legal time. Additional schema attributes can be used to indicate if the display of the time should be shifted to the legal time of a different time zone. For example, if the data is stored in the base time zone but the data is related to a different time zone, the data will be shown in the time zone appropriate for the data (including the appropriate seasonal adjustment). Refer to Schema Nodes and Attributes- Standard Time Considerations for more information.

Exchanging Date / Time Information With External Systems

Date/time information should be exchanged between systems in standard XSD format as it includes a reference to the time zone of the specified time as an offset from Coordinated Universal Time (UTC). The system automatically converts date/time elements in an inbound message from XSD to internal format and vice versa for an outbound message. The latter is controlled by an explicit date/time format setting on the External System record for the outbound message type.

The offset is determined as followed and it is based on the time zone associated with the element as explicitly defined by the respective inbound or outbound schema:

- Elements defined as stored in the standard time of the base or other time zone would always have the same offset throughout the year as this time line is never shifted for daylight saving. The offset would be the standard offset of that time zone from UTC, i.e. not during daylight saving time.
- Elements stored in the legal time of the base time zone would have the standard offset of the base time zone when the date/time value is outside the daylight saving period and the shifted offset when it is inside that period of time.

User Time Zone

If your company does business in multiple time zones, the user record of each user may reference the time zone of their location. This time zone provides additional information about the location of the user and as such may be used by specific business rules that involve time zone settings.

Note: The time zone defined on the user record is not used to automatically display date/time information in the user's time zone. By default, the system displays date/time fields that are stored in standard time in the legal time of their respective time zone and not the user's time zone.

Setting Up Time Zones

Refer to Designing Time Zones for background information about defining time zones.

Open this page using **Admin** > **General** > **Time Zone** > **Search** .

Copyright © 2024, Oracle and/or its affiliates.

This is a standard All-in-One portal.

Refer to the embedded help text for more information on the Time Zone attributes.

Setting Up Seasonal Time Shift

Note: The information in this topic applies only to Oracle Utilities Customer Care and Billing - Interval Billing applications.

Open **Admin** > **General** > **Seasonal Time Shift** > **Search** to define the seasonal time shift schedule.

Description of Page

Enter a unique **Seasonal Time Shift** code and **Description** for the seasonal time shift.

The Collection defines the **Effective Date/Time** (in standard time) that a time zone may shift in and out of standard time. If time is changed from standard time on the effective date/time, enter the **Shift in Minutes** that the time changes from standard time (usually **60**). If the time is changed back to standard time on the effective date/time, enter a **Shift in Minutes** of **0**.

Defining Geographic Types

If your company uses geographic coordinates for dispatching or geographic information system integration, you need to setup a geographic (coordinate) type for each type of geographic coordinate you capture on your premises and/or service points (geographic coordinates can be defined on both premises and service points).

Note: Product specific. There is no framework functionality that uses this information. Refer to your specific product documentation to verify how this table is used in your specific product. In addition, you can determine if this object is a foreign key on any tables specific to your product. Navigate to the Table page in the application, search for and select the Geographic Type table and use the Relationships tab to view this information.

To define geographic types, open **Admin** > **Geographic** > **Geographic Type.**

This is a standard All-in-One portal.

The **Validation Format Algorithm**, if specified, is used to validate that the geographic location entered on the premise and/or service point for the geographic type is in the format as defined in the algorithm. If you require validation, you must set up this algorithm in the system.

Defining Work Calendar

This portal is used to view and maintain information about a work calendar. Work calendars are used to ensure system-calculated dates fall on a workday.

You can access the portal from **Admin** > **General** > **Work Calendar**. You are brought to a query portal with options for searching for a specific calendar. Once a calendar has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones will appear as part of the portal's **Main** tab page

- **Work Calendar.** This zone provides information about the selected calendar.
- **Work Calendar Holidays.** This zone lists the holidays observed by your organization for this calendar. Use the zone actions to add, maintain or duplicate a work holiday entry.

Copyright © 2024, Oracle and/or its affiliates.

Defining Display Profiles

When you set up your users, you reference a display profile. A user's display profile controls how dates, times, and numbers displayed. Choose **Admin** > **General** > **Display Profile** > **Search** to maintain display profiles.

Description of Page

Enter a unique **Display Profile** and **Description** to identify the profile.

Enter a **Date Format**. This affects how users view dates and how entered dates are parsed. The following table highlights standard supported date mnemonics and what is displayed at runtime.

Mnemonic	Comments
dd	Day of the month.
d	Day of the month, suppressing the leading 0.
MM	Month number.
M	Month number, suppressing the leading 0.
yyyy	The 4-digit year.
yy	The 2-digit year.
y	Allows entry in either 2 or 4-digit form and is displayed in 2-digit form.

Other characters are displayed as entered. Typically, these other characters should be separators, such as "-", ".", or "/". Separators are optional; a blank space cannot be use.

Examples:

Configuration Format	Sample Output
MM-dd-yyyy	04-09-2001
d/M/yyyy	9/4/2001
yy.MM.dd	01.04.09
MM-dd-y	04-09-01 - In this case you could also enter the date as 04-09-2001

Note: For centuries, the default pivot for 2-digit years is **80**. Entry of a 2-digit year greater than or equal to **80** results in the year being interpreted as 19xx. Entry of a 2-digit year less than **80** results in the year being interpreted as 20xx.

In addition, the following date localization functionality is supported. Note that in every case, the date is stored in the database using the Gregorian format. The settings below result in a conversion of the date for the user interface.

- **Hijri Dates**

  Entering **iiii** for the year is interpreted as a year entered and displayed in Hijri format. For example, the Gregorian date 2014–05–30 may be entered / displayed as 1435/07/30 for a user whose display profile date format is **iiii/MM/dd**. Note that this functionality relies on date mapping to be defined in the Hijri to Gregorian Date Mapping master configuration. entry. Refer to Additional Hijri Date Configuration for more information.
- **Taiwanese Dates**

  Entering **tttt** for the year is interpreted as a year entered and displayed in Taiwanese format where year 1911 is considered year 0000. For example, if the Gregorian date is 01-01-2005, it is displayed as 01-01-0094 for a user whose display profile date format is **dd-mm-tttt**.
- **Japanese Dates**

Copyright © 2024, Oracle and/or its affiliates.

There are two options available for configuring Japanese Era date support. The setting **Gyy** for the year is interpreted as a year entered and displayed using an English character for the era followed by the era number. The letter 'T' is used for dates that fall within the *Taisho* era. The letter 'S' is used for dates that fall within the *Showa* era. The letter 'H' is used for dates that fall within the *Heisei* era. The letter 'R' is used for dates that fall within the *Reiwa* era. For example, for a user whose display profile date format is **Gyy/mm/dd** the Gregorian date 2008/01/01 is shown as **H20/01/01**; the Gregorian date 1986/03/15 is shown as **S61/03/15**. The setting **GGGGyy** is interpreted as a year entered and displayed using Japanese characters for the era followed by the era number.

Japanese date limitations are as follows:

- The years 1912 through the current date are supported.
- Any functionality that displays Month and Year does not support Japanese Era dates. These dates are shown in Gregorian format.
- Graphs that display dates do not support the **GGGGyy** format.

Enter a **Time Format**. The following table highlights standard supported date mnemonics.

Mnemonic	Comments
hh	The hour 1-12.
h	The hour 1-12, suppressing the leading 0.
HH	The hour 0-23.
H	The hour 0-23, suppressing the leading 0.
KK	The hour 0-11.
K	The hour 0-11, suppressing the leading 0.
kk	The hour 1-24.
k	The hour 1-24, suppressing the leading 0.
mm	Minutes.
m	Minutes, suppressing the leading 0.
ss	Seconds.
s	Seconds, suppressing the leading 0.
a	Indicates to include **am** or **pm**. This is only needed for 12 hour formats, not 24 hour formats. (**hh**, **h**, **KK**, **K**). If an **am** or **pm** is not entered, it defaults to **am**.

Examples:

Configuration Format	Sample Output
hh:mma	09:34PM (can be entered as 09:34p)
hh:mm:ss	21:34:00
h:m:s	9:34:0

There are several options for displaying Numbers.

Decimal Symbol defines the separator between the integer and decimal parts of a number. Valid values are "**.**" (a period) or "**,**" (a comma).

Group Symbol defines the means to separate groups of bigger numbers. Valid values are as follows:

- A comma ("**,**"). Large numbers group by threes separated by a comma, for example 1,000,000.
- A period ("**.**"). Large numbers group by threes separated by a period, for example 1.000.000.

Copyright © 2024, Oracle and/or its affiliates.

- **None**. Large numbers do not have any separator, for example 1000000.
- **South Asian**. This option uses a comma for its separator but will group large numbers as follows: the first comma is used for the thousands separation and numbers over 9,999 are grouped with 2 units, for example 10,00,000.
- **Space**. Large numbers group by threes separated by a space, for example 1 000 000.

Negative Format defines how negative values are displayed. Valid values are **-9.9**, **(9.9)**, or **9.9-**.

Currency values can have a different **Negative Format** from other numbers. Valid values are **-S9.9**, **(S9.9)**, or **S9.9-**, where the "S" represents the currency symbol.

Additional Hijri Date Configuration

For implementations that wish to support displaying dates according to the Hijri calendar, besides appropriate configuration in the Display Profile, the mapping between the Hijri dates and the Gregorian dates must be entered. This mapping is defined in the **Hijri to Gregorian Date Mapping** master configuration record.

The mapping record contains a collection of entries for each year in the Islamic calendar.

For each year, clicking the Expand Zone icon shows the mapping collection with the first date of each month of the Hijri calendar. The corresponding date in the Gregorian calendar should be entered for each row.

Defining Phone Types

Phone types define the format for entering and displaying phone numbers.

To add or review phone types, choose **Admin** > **General** > **Phone Type.**

This is a standard All-in-One portal.

The **Phone Number Format Algorithm** for each phone type controls the format for entry and display of phone numbers.

Use **Phone Type Flag** is used to identify phone types that may have specific functionality associated with it. For example if your product had support for sending a facsimile transmittal, identify the phone type that captures a **Fax** number.

Defining Feature Configurations

Some system features are configured by populating options on a "feature configuration". Because various options throughout the system may be controlled by settings in feature configuration, this section does not document all the disparate possible options. The topics below simply describe how to use this transaction in a generic way.

For information about specific features:

- Refer to the detailed description of each option type.
- Use the index in the online help and search for ‘feature configuration’ to find any specific topics describing feature options in the administration guide.

You can create options to control features that you develop for your implementation. To do this:

- Review the lookup values for the lookup field **EXT_SYS_TYP_FLG**. If your new option can be logically categorized within an existing feature type, note the lookup value. If your new option warrants a new feature type, add a lookup value to this lookup field.
- Define the feature's option types. If you have identified an existing feature type to add the options to, find the lookup with the name **xxxx_OPT_TYP_FLG** where **xxxx** is the lookup value of **EXT_SYS_TYP_FLG** noted above. If you decided to create a new feature type (by adding a new lookup value to the **EXT_SYS_TYP_FLG** lookup, you must create a new lookup with the name **xxxx_OPT_TYP_FLG** where **xxxx** is the new value you defined above.
- Flush all caches.

Copyright © 2024, Oracle and/or its affiliates.

Feature Configuration - Main

To define your feature configuration, open **Admin** > **General** > **Feature Configuration.**

Use the **Feature Configuration Query** portal to search for an existing feature configuration. Once a feature configuration is selected, you are brought to the maintenance portal to view and maintain the selected record.

Description of Page

Enter an easily recognizable **Feature Name** code.

Indicate the **Feature Type** for this configuration. For example, if you were setting up the options for the external messages, you'd select **External Messages**.

Note: You can add new Feature Types. Refer to the description of the page above for how you can add Feature Types to control features developed for your implementation.

Note: Multiple Feature Configurations for a Feature Type. Some Feature Types allow multiple feature configurations. The administration documentation for each feature will tell you when this is possible.

The **Options** grid allows you to configure the feature. To do this, select the **Option Type** and define its **Value**. Set the **Sequence** to **1** unless the option may have more than value. **Detailed Description** may display additional information on the option type.

Note: Each option is documented elsewhere. The administration documentation for each feature describes its options and whether an option supports multiple values. Use the index to look for 'feature configuration' to find the various types of feature options.

Note: You can add new options to base-package features. Your implementation may want to add additional options to one of the base-package's feature types. For example, your implementation may have plug-in driven logic that would benefit from a new option. To do this, display the lookup field that holds the desired feature's options. The lookup field's name is **xxxx_OPT_TYP_FLG** where **xxxx** is the identifier of the feature on the **EXT_SYS_TYP_FLG** lookup value. For example, to add new batch scheduler options, display the lookup field **BS_OPT_TYP_FLG**.

Feature Configuration - Messages

If the feature exists to interface with an external system, you can use this page to define the mapping between error and warning codes in the external system and our system.

Open this page using **Admin** > **General** > **Feature Configuration** and navigate to the **Messages** tab.

Description of Page

For each message that may be received from an external system, define the **Feature Message Category** and **Feature Message Code** to identify the message.

A corresponding message must be defined in the system message tables. For each message identify the **Message Category** and **Message Number**. For each new message, the Message Category defaults to **90000** (because an implementation's messages should be added into this category or greater so as to avoid collisions during upgrades).

Defining Master Configurations

A master configuration is an object that enables an implementation to define configuration for features in the system. It is an alternative to using feature configuration for defining options. A master configuration is defined using a business object. Only one master configuration may exist for a given business object.

Copyright © 2024, Oracle and/or its affiliates.

Overview / Initial Setup

The product provides one or more master configuration that may be used for configuration. Some examples of base master configuration business objects are as follows

- **Hijri to Gregorian Date Mapping**. This allows an implementation that uses Hijri dates to define the mapping between those dates and Gregorian dates.
- **ILM Configuration.** For implementations that use Information Lifecycle Management, the ILM configuration record defines some parameters used by the process.
- **Migration Assistant Configuration.** For implementations that use the content migration assistant (CMA), the configuration record defines some parameters used by the process.

For a list of all the master configuration records provided by the product, navigate to the master configuration page in the application. To find help topics related to functionality controlled by the master configuration records, use the keyword ‘master configuration’ in the index.

Implementations may configure the business objects to limit the ones that are visible to users if desired.

- If the master configuration does not apply to your implementation, navigate to the business object for this master configuration and update the Instance Control to **Do not allow new instances**. This ensures that the business object will not be visible to any user for the implementation.
- If a given master configuration record is only relevant for certain users in the system, application security may be used to limit the records visible by a given user. By default the base delivered business objects typically reference the Master Configuration MO application service. This may be overridden to link custom application services to the master configuration BOs to provide more granular security. Users will only see the master configurations that they have security for.

Configuration

To set up a master configuration, open **Admin** > **General** > **Master Configuration.**

The topics in this section describe the base-package zones that appear on the Master Configuration portal.

Master Configuration

The Master Configuration List zone lists every category of master configuration.

The following functions are available:

- If a master configuration record exists for a given master configuration business object, the broadcast icon may be used to view details information about the adjacent master configuration. In addition, an edit icon is visible to allow a user to update the record.
- If a master configuration record does not exist for a given master configuration business object, the add icon is visible to allow a user to define the record.

Master Configuration Details

The Master Configuration Details zone contains display-only information about a master configuration.

This zone appears when a master configuration has been broadcast from the Master Configuration zone.

Please see the zone's help text for information about this zone's fields.

Copyright © 2024, Oracle and/or its affiliates.

Chapter 2

Security & User Options

Topics:

- Application Security
- Row Security
- Data Privacy
- Cryptography Keys

The contents of this section describe how to maintain a user's access rights.

Application Security

The system restricts access to various components of the application using application services. The following sections provide more information about this mechanism.

Understanding Application Security

The contents of this section and supporting topics provide information about application security.

Users

Each user who is granted access to the application is stored in the User object. This object defines the attributes of the user including identifier, name, language, time zone, home page and display profile (which defines the preferred format of dates and amounts). In addition there is configuration related to using the product, including favorite links and scripts, portal preferences and valid To Do roles - controlling what types of To Do entries a user can work on. Additionally, Refer to Users and User Groups for additional information.

Application Services

Each object in the system that may be secured is associated with an application service. For example, each page in the system defines its application service allowing you to control what users can view a given page.

An application service defines one or more Acces Modes, which typically correspond to an action associated with the application service. This allows you to control users that can only view a type of record (and grant security to the Inquire access mode) and users that can perform other actions such as add, change and delete. Refer to Secured Objects and Application Services to understand what objects are associated with an application service and the typical access modes associated with each type of secured object.

Application Services may also define additional configuration, called security types. Refer to Field Level Security for more information.

User Groups

User Groups are a mechanism for grouping users, usually around job roles. Users are attached to User Groups through a relationship that is effective-dated. This can be useful for the attachment of temporary employees such as contractors or for people who change roles regularly.

Each User Group is authorized to access appropriate Application Services and their access modes. In addition, user groups define access to the application service's security types, if applicable. Refer to Field Level Security for more information.

Users and User Groups

A user group defines the application services and access modes its users have access to. A user may be associated to more than one user group. The user groups may have overlapping security configurations and the security access for a user is the union of all security granted by all of its user groups.

A major effort for implementations is to decide how to define their user groups. User groups can be created to represent roles or other job-related categorization; or based on organizational factors; or a combination of both. Refer to Secured Objects and Application Services for some guidelines on understanding what application services secure and tools provided to help security administrators identify what application services to grant access to.

The following sections provide additional topics related to users and user groups, including some base functionality related to these objects as well as some best practices.

Copyright © 2024, Oracle and/or its affiliates.

Base Delivered User / User Groups

The system is delivered with a base User and one or more base User Groups. They are provided to facilitate an initial user accessing the application.

The system supplied user is **SYSUSER**. This user is delivered so that when an environment is provisioned, there is at least one user authorized to use the system. It has access to the base delivered user groups. An initial administrator can sign in as this user and proceed to create the actual users that should get access to the environment.

Important: Once appropriate users have been defined, it is recommended to update SYSUSER to set it to disabled.

All active application services that are provided by the product are linked to a base delivered user group. The supported application services delivered by Oracle Utilities Application Framework are linked to user group **ALL_SERVICES** with all their supported access modes. Other products in your application stack may include their base delivered application services in this user group. Additionally, your specific product may deliver specialized user groups that group specific application services for a given functional area. You cannot change or remove the information delivered for the base user groups. System administrators should define user groups appropriate for their business requirements.

Important: It is not recommended for any production user to have access to the base delivered user groups.

Defining a Read-Only User Group

The product does not supply a user group that is 'read only', but it is a common requirement for implementations. This section provides a guideline for creating such a user group.

1. Create the User Group with the desired name and description.
2. Navigate to the Service Manager tab. In the Add Application Service / Access Modes tab, you can use the filter to limit the results to those with Inquire access mode. Click the checkbox in the header to select all rows and click Add. You are prompted for an expiration date. Click Save. (Note that the zone has a limit to 1000 services so you may need to do this more than once to get all services added.)
3. In addition, you should give this user group access to all the access modes for the default application service: **F1-DFLTS** (Default Application Service). This application service is considered one that all users should have for basic access to the system.

Template User

The system provides support for defining one or more users as a template user. There are two broad use cases for using a template user.

Inheriting Preferences or Favorites Configuration

Some implementations want all users of a certain team to have the same configuration of portal preferences and / or favorite links. This allows users in that team to have the same information in the same place on their respective displays, facilitating in helping each other with questions.

If your organization wants to implement this practice, define a template user with the desired portal preferences and / or favorite links. You may define different users with different configurations for more granularity.

For each user whose preferences or links should be taken from the template user, go to the User record via **Admin** > **Security** > **User.** If the portal preferences should be inherited from a template user, define the desired template user as the **Portals Profile User** . If the favorite links should be inherited from a template user, define the desired template user as the **Favorites Profile User**.

If applied, the specified user's portal preferences and / or favorite links will be those defined on the template user and they will be unable to customize those preferences.

Defining a Template User to help in User Creation

The user object in the product captures configuration related to many aspects of using the product. It includes application security configuration, such as its user groups; row level security configuration through its access groups; the To Do roles that define what types of To Do entries a user can work on, along with other configuration. Many users that are in

Copyright © 2024, Oracle and/or its affiliates.

a similar 'role' in the organization would have many of the same settings. As such, it may be useful for implementations to define a template user that represents a 'role' in the organization, with much of the configuration common for users in that role. When creating a new user, you can use the template user as the starting point and then duplicate that record to create the new user. Then you only need to adjust the information that is unique for the newly created user.

Many implementations provision their users first in an external identity management system and use an interface to import the user definition into the product. Refer to Importing Security Configuration from an External Source for more information. For these interfaces, the external system may have the ability to define custom information relevant to the product's user record, but the product also supports referencing a template user so that information from the template user is copied to the new user while creating the record.

How can you refer to a template user when creating a User from and external source?

- When the new user is uploaded to the system, the interface uses the user BO **F1–IDMUser** to create the user. The BO includes a pre-processing algorithm. This algorithm looks for the existence of a template user in the schema of the new user sent as a characteristic of type **F1-TMUSR**. If such a characteristic is found, it uses that template user. If not found, the algorithm supports defining a template user as an algorithm parameter. If a template user is identified, all the information from the template user will be copied onto the new user record except for the information passed in from the external source. The template user is captured on the newly created user via a characteristic for information / audit purposes.

  Note: Bookmarks are not included in the data that is copied from a template user.

- Once the new user is created, its configuration can now be adjusted, if applicable. Note that the template user is only a tool used when adding a user. Updates to the template user will not "ripple" to all the other users that were created based on this template.
- Not all external sources may have the ability to define a template user and send it as a characteristic or via another field. For these integrations, using a single template user on the algorithm as a parameter may be used to define basic configuration applicable to a large percentage of new users. (More information on this below). For those integrations that are able to associate a new user with a template user in the external system and provide that template user along with the new user information allows you to associate the new user with a template user that is more aligned with the user's role. Refer to Importing Security Configuration from an External Source for more information on how each integration supports the definition of a template user.

Designing your Template Users for helping Create Users

Before configuring template users, all the administrative control tables that are part of User configuration must be defined, including time zones, display profiles, To Do roles, data access roles and user groups.

If your integration will use a single template user defined as an algorithm parameter, you need to define a template user with the basic information that is applicable to all or at least most users. See below for guidance on referencing this template user as an algorithm parameter.

If the way you create users allows you to associate the user with a more specific template user, the next step is to define the user configuration for your system users. During this exercise, you will find that you have broad categories of users. But you will also see that within a given category of user there may be variations in the user privileges and preferences. For example, perhaps there are supervisors within the Customer Service role that have more security privileges than a typical Customer Service user. In addition, there may be variations based on the attributes of the users themselves. For example, maybe your organization exists in multiple time zones and some of your workers are in one time zone and some are in the other.

At this point your security users that are designing their user provisioning procedures must decide the following:

- What information about a new user will be captured in the identity system (besides the expected information like Name, Email and the User IDs)? For example, for the case of multiple time zones, maybe the best solution is to capture the time zone when defining the user.

  What information is defined on the template user and how many template users should be created to reduce the need for manual steps or additional data captured in the identity management system? In the case of multiple time zones, proliferating the template users to have one set for one time zone and another set for the other time zone may not make sense since this is one field that is different. However it may be reasonable to create additional templates in the case of striations in the levels of privileges for workers of a different category. So rather than template users

Copyright © 2024, Oracle and/or its affiliates.

for Dispatcher, Mobile Worker, System Administrator and Contractor, your organization may have template users for Dispatcher, Mobile Worker, Mobile Worker (Supervisor), System Administrator, Contractor (Short Term) and Contractor (Long Term).

What information is must be configured on the User record in the application after the user is added? If only a small number of users have a variation from other users, it may be that the easiest way to deal with those variations is to simply update those user records manually. Using the above examples:

- If your organization covers 2 time zones but only a small group of people work in one of the time zones whereas the bulk of the users are in the other time zone, the simplest procedure may be to define the template users for the main time zone and use that for the creation of all users. Then for the small group of users in the separate time zone, navigate to the User page to adjust the time zone after the record is added.
- If only a small number of Mobile Workers are supervisors with separate privileges, rather than defining a special template user for those type of workers, the simpler procedure may be to use the Mobile Worker template and then navigate to the User page to add the additional privileges to the supervisor users after the record is added.

Defining a Template User

Regardless of which use case is applicable for your organization, the procedure for defining a template user is the same.

Navigate using **Admin** > **Security** > **User.**

- Define a User ID. If you are using a template user to help create users from an external system and the external system is able to capture the template user to send in with the new user information, this is the ID that should be referenced in the external system.
- Be sure to choose a **User Type** of **Template User**.
- Define all the information appropriate based on the use case.
  - For template users used as a Portal Profile user, be sure to define the appropriate portal profile information.
  - For template users used as a Favorites Profile user, be sure to define the appropriate favorite links.
  - For template users acting as a type of 'role' to help with creation of users, define all the information that should be copied onto a new user that references this user as its template user. (Note that **Bookmarks** are not included in the data that is copied from a template user.)

Defining a Default Template User for Integrations

Define a default template user on the algorithm the defaults data from the this user as follows.

- Navigate using **Admin** > **System** > **Algorithm.**
- Search for and select the algorithm **F1-OIMUSR** (Populate User Data from a "Template" User).
- Edit the record. Add a new row in the Algorithm Versions collection. Enter an appropriate date (later than the date of the existing entry, but on or before the current date). Populate the template user you want to use as the default.

Secured Objects and Application Services

The system restricts access to transactions or explicit services using an application service. The following sections provide more information about what objects are secured, how to find out what is secured by each app service as well as additional information related to managing application services.

What is Secured by an Application Service

- The following points highlight security related to viewing and modifying individual records in the system:
  - All maintenance objects define an application service that includes the basic actions available, typically **Add**, **Change**, **Delete**, and **Inquire**. The base product supplies an application service for every maintenance object. Some objects that include a fixed lifecycle, may define additional access modes on the maintenance object's application service. The application service for the maintenance object is defined on its related service program.

Copyright © 2024, Oracle and/or its affiliates.

> **CAUTION:** Important! If an application service supports actions that modify the database other than **Add**, **Change**, and **Delete**; you must provide the user with **Change** access in addition to the other access rights. Consider a maintenance object that includes access modes **Freeze**, **Complete** and **Cancel**). If you want to give a user access to any of these special actions, you must also give the user access to the **Change** action.

- For maintenance objects whose user interface page is not portal-based, the application service also controls whether the menu entry appears. If a user doesn't have access to the maintenance object's application service, the menu item that corresponds with the application service will not be visible.
- For portal based user interfaces, each main (stand-alone) portal defines an explicit application service with the access mode **Inquire**, allowing the user interface to be secured independently of the underlying object security. If a user doesn't have access to the portal's application service, the menu item that corresponds with the application service will not be visible. The base product supplies an application service for every portal that is accessible from the menu. Note that the application service for the portal is defined on its related service program, which is derived via its navigation option and navigation key.
- Menu items may define an application service / access mode. Typically the security supplied for portals and maintenance objects provides enough granularity to suppress menu items that a user does not have access to. In addition, by default, the system populates application services / access modes on menu items related to a BPA script. A common use case is the 'Add' menu item for maintaining an object. In this case, typically the application service for the maintenance object is used along with the **Add** access mode. Other types of BPA related menu items may have other appropriate configuration.

  Clients may choose to linking an explicit application service / access mode to further suppress a menu item. For example, imagine your implementation creates a special BPA script to add a To Do Entry and would like users to use the special BPA rather than the base supplied Add dialogue for To Do Entry. The base delivered menu item for adding a To Do Entry uses the To Do Entry maintenance object application service with the Add access. Users need that security setting to be able to add a To Do Entry even from a different dialogue. To suppress the base Add dialogue, override the setting for the base supplied menu item for To Do Entry Add with a special application service and access mode. Then define a menu entry for the new special BPA for adding.
- Zones define an application service.
  - For zones linked to a portal, if a user doesn't have access to the zone's application service, the zone will not be visible on the portal. In most cases the zone is delivered with the same application service as its portal. In special cases, such as the zones on the Sidebar, the product supplies separate application services for each zone allowing implementations to determine at a more granular level which users should have access to which zones.
  - For query zones that are configured on a multi-query zone, if a user doesn't have access to the zone's application service, the zone will not be visible in the dropdown on the multi-query zone. In most cases all zones in a multi-query zone define the same application service as the multi-query zone. The product may supply a special application service for one or more zones in a multi-query zone if the functionality is special to certain markets or jurisdictions and not applicable to all implementations.
  - For zones that are used by business services to perform SQL queries, the product supplies a default application service (**F1-DFLTS**). Security for these zones is not checked by the product as they are used for internal purposes.
- For portal based pages, individual elements may be shown or hidden based on security using the **oraHasSecurity** function in a UI map's HTML or in UI Hints in a schema. Refer to Check User's Security Access for more information.
- Business objects define an application service. If the business object defines a lifecycle, the application service must include access modes that correspond to each state. In addition, the standard maintenance object access modes of **Add**, **Change**, **Delete** and **Inquire** are included. The base product business objects are supplied with appropriate application services. In addition, implementations may override the configured application service if desired.

> **CAUTION:** Important! If an application service supports actions that modify the database other than **Add**, **Change**, and **Delete**; you must provide the user with **Change** access in addition to the other access rights. Consider a business object that includes lifecycle states **Freeze**, **Complete** and **Cancel**). If you want to give a user access to any of these special actions, you must also give the user access to the **Change** action.

Copyright © 2024, Oracle and/or its affiliates.

  - Batch controls define an application service which provides the ability to secure submission of individual batch processes. The application service must include an access mode of **Execute**. The base product batch controls are supplied with appropriate application services. These services will typically have an ID that matches the batch control ID.
  - Report Definition records define an application service. The application service must include an access mode of **Submit / View Report**.
- The following objects are securable but are typically executed via internal processes. The security is provided to ensure that any access to the objects from an external source is secured.
  - BPA scripts may define an application service with the access mode **Execute**. The base BPA scripts are typically not configured with any application service. An implementation may define one. Note that as mentioned above, a menu item may also be configured with an application service and access mode. This allows for a BPA that is invoked via a menu entry to be secured in more than one way.
  - Business Services and Service Scripts define an application service with the access mode **Execute**. This is needed for services that may be executed from an external system, for example via an inbound web service. Base business services and service scripts that are linked to an inbound web service are configured with special application service. All other business services and service scripts are delivered with a default application service (**F1-DFLTS**), which may be overridden by an implementation.
  - Scripts may also check security within an Edit Data step type if there is some functionality that should or should not execute based on a user's security access. The Edit Data step would include a call to the business service **F1-CheckApplicationSecurity**
  - Service Programs define an application service. As mentioned above, for Portals and Maintenance Objects, their application service is taken from the related service program. In base, specific application services are released for each of these types of service programs. All other service programs are typically delivered with a default application service (**F1-DFLTS**), which may be overridden by an implementation. Note that for service programs linked to a Business Service, the application service on the business service takes precedence when invoking the business service.

Determining What an Application Service Secures

Because the product wants to provide as much granularity as possible so that implementations can have the greatest control over what users can do, there are a large number of application services provided out of the box and it can be daunting to understand who needs to have access to what application services to do their job. The product provides several online tools to help security administrators to understand what application services exist and what they are for.

- On the application service page itself, there is a Secured Objects zone that indicates all the metadata that is related to this application service. There are a small number of application services that are securing functionality that is not discoverable by the Secured Objects zone. For those services, the detailed description of the application service should indicate what it's for.
- A large percentage of security is related to a specific page in the system that is accessible via the menu. Because of the different layers of security, to be able to view a record you may need access to several application services: the one associated with the query portal, the one associated with the maintenance portal, the one associated with the maintenance object of the record and the one associated with the business object governing the record. When viewing a menu, there is an Application Security tab, which allows you to review each menu line and see a list of all the application services a user may need access to for viewing or maintaining data on that related page.
- The Application Security Query portal provides various filters for viewing what objects are being secured by what services. Besides menu items, it shows batch controls, reports, web services and other functionality. In addition, it allows you to view what users and what user groups currently have access to what application services and allows you to compare the security of two users.
- The User Group portal includes several zones that help security administrators to configure security options or review security configuration. The main tab includes a zone to Add Application Services. This zone only shows application services that are not already linked and provides several ways to narrow down the list of application services. The Service Manager tab has a similar zone, but breaks down the information to a combination of application service and access mode. Finally, the Secured Components tab is similar to the Application Security Query. It allows you to query the objects secured by application services in this user group.

Copyright © 2024, Oracle and/or its affiliates.

Application Services Introduced in a Release

When new functionality is provided that includes any object that is associated with an application service, new application services may be introduced. The product release notes indicate the new application services included in the release. In addition, the product includes a special Entity Tag that includes a list of the new application services. The tag's description uses the cloud release reference rather than the on-premise release reference to identify itself.

Field Level Security

Sometimes transaction and action security is not sufficient. There are situations where you may need to restrict access based on the values of data. For example, in Oracle Utilities Customer Care and Billing you might want to prevent certain users from completing a bill for more than $10,000. This is referred to as "field level security".

Field level security can be complex and idiosyncratic. Implementing field level security always requires some programming by your implementation group. This programming involves the introduction of the specific field-level logic into the respective application service(s). The recommendation is to use a business object validation algorithm for checking field level security.

> **Note:**
>
> The check for field level security must exist in validation logic to ensure that data is not added or updated if it does not pass the field level security rules.
>
> It is not straight forward to implement user interface changes related to field level security, such as protecting elements under certain conditions. This is because overriding the base product delivered user interface to check for implementation specific conditions is not always supported, depending on how the user interface is built, or requires a full duplication of product artifacts, which is not recommended.

The definition of a user's access rights is performed using the same transactions used to define transaction / action level security. This is achieved as follows:

- Create a security type for each type of field-level security.
- Define the various access levels for each security type. For example, assume you have some users who can complete bills for less than $300, and other users who can complete bills for less than $1,000, and still other users who can complete bills for any value. In this scenario, you'd need 3 access levels on this security type:
  - Level 1 (lowest): May authorize bills <= $300
  - Level 2 (medium): May authorize bills <= $1,000
  - Level 3 (highest): May authorize all bills
- Link this security type to each application service where this type of field level security is implemented. This linkage is performed on the security type transaction.
- Defining each user group's access level for each security type (this is done for each application service on which the security type is applicable).

> **Note:**
>
> Highest value grants highest security. The system expects the highest authorization level value to represent highest security level. Moreover, authorization level is an alphanumeric field so care should be taken to ensure that it's set up correctly.

Importing Security Configuration from an External Source

The product provides support for importing security information from an external source. In all cases, the product uses a special user BO **F1–IDMUser** to create the user and the recommendation is to take advantage of the Template User functionality. More information about the out of the box support for using this feature is described in the related links.

- If your organization uses Lightweight Directory Access Protocol (LDAP), you can import your existing LDAP users and groups into the system. Once imported, all user and group functions are available. You can import a user group, or a single user. You can resynchronize your LDAP users and groups at any time.

Copyright © 2024, Oracle and/or its affiliates.

> **Fastpath:** For more information refer to LDAP Integration.

- The system provides an integration with Oracle Identity Manager. When a user is created in the identity manager product, its information can automatically be interfaced to the product. Once the user is successfully created in the system, all functions are available.

> **Fastpath:** For more information refer to Oracle Identity Manager Integration.

- The system provides a web service with add, update and get operations that follow the System for Cross-domain Identity Management (SCIM) specification for provisioning users.

> **Fastpath:** For more information refer to Provision Users with SCIM Open Standard.

Defining Application Services

Please refer to Understanding Application Security for a description of how application services are used when you grant user groups access rights to transactions and the significance of an application service's access modes.

You can access the portal using **Admin** > **Security** > **Application Service.** You are brought to a query portal with options for searching for a specific application service. Once a record has been selected you are brought to the maintenance portal to view and maintain the selected record.

> **Note:** When introducing a new record, carefully consider its naming convention. Refer to System Data Naming Convention for more information.

The following zones may appear as part of the portal's **Main** tab page

- The **Application Service** zone provides the information about the application service and its access modes.
- The **Secured Objects** zone displays the object (or objects) that are secured by this application service. Refer to Secured Objects and Application Services for details about the types of objects that may be secured.

The **Application Security** tab includes zones to view and manage which user groups have access to the application service and which ones do not have access .The following zone may appear as part of this tab page:

- The **User Groups Linked** zone lists the user groups that currently have a link to the application service. Note that expired links are included when the zone is first displayed. Use the zone filters to exclude expired links and to limit the user groups shown by user ID, user group ID, and user group description.
- The **User Groups not Linked** zone lists the user groups that do not have a link to the application service. Use the zone filters to limit the user groups shown by user ID, user group ID, and user group description.

Defining Security Types

Security types are used to define the types of field level security.

> **Note: Programming is required.** You cannot have field level security without introducing programming code. Refer to Field Level Security for more information on how security types are used to define field level security.

Security Type - Main

Select **Admin** > **Security** > **Security Type** to define your security types.

Description of Page

Enter a unique **Security Type** and **Description**.

Use the **Authorization Level** grid to define the different authorization levels recognized for this security type. Enter an **Authorization Level Number** and its **Description**.

Copyright © 2024, Oracle and/or its affiliates.

Note: Programming is required. Note that the values that you enter are not interpreted by the system itself, but by the user exit code used to implement the special security. Check with the developer of the user exit logic for the correct values. Refer to Field Level Security for more information on how security types are used to define field level security.

Use the **Application Services** grid to define the application service(s) to which this security type is applicable. If this application service is already associated with user groups, you must update each user group to define their respective security level. This is performed on the User Group portal.

Defining User Groups

A user group is a group of users who have the same degree of security access. Think of a user group as a "role"; associated with a role are:

- The users who play this role
- The application services to which the role's users have access (along with the actions they can execute for each service and their field level security authorization levels).

Refer to Secured Objects and Application Services and Field Level Security for more information about access modes and security types.

CAUTION: Important! If an application service supports actions that modify the database other than **Add**, **Change**, and **Delete**; you must provide the user with **Change** access in addition to the other access rights. Consider a transaction that supports actions in addition to **Add**, **Change**, and **Inquire** (e.g., **Freeze**, **Complete**, **Cancel**). If you want to give a user access to any of these additional actions, you must also give the user access to the **Inquire** and **Change** actions.

This portal allows a security administrator to view and manage information about a user group.

You can access the portal from the**Admin Menu** > **Security** > **User Group**. You are brought to a query portal with options for searching for a specific user group. Once a user group has been selected you are brought to the maintenance portal to view and maintain the selected record.

CAUTION: Application services may not be changed or removed from the **ALL_SERVICES** user group. Refer to Base Delivered User / User Groups for an explanation.

The following zones may appear as part of the portal's **Main** tab page

- **User Group.** This zone provides information about the selected user group.
- **Application Services.** This zone lists application services linked to the selected user group. You may use this zone to edit the information for an specific application service or use mass actions to remove selected services or update their effective date.
- **Add Application Services.** You may use this zone to add application services to the selected user group.

The following zones may appear as part of the portal's **Users** tab page

- **Users.** This zone lists users linked to the selected user group. You may use mass actions to remove selected users or update their effective date.
- **Add Users.** You may use this zone to add users to the selected user group.

The following zones may appear as part of the portal's **Service Manager** tab page

- **Application Service / Access Modes.** This zone lists application service and access mode combinations currently linked to the selected user group. You may use mass action to remove a selected set of records.
- **Add Application Service / Access Modes.** You may use this zone to add application service access mode records.
- **Security Type Authorization.** This zone maintains field level security for application services included in the selected user group.

The following zone may appear as part of the portal's **Secured Components** tab page

- **Application Components.** This zone allows you to review various types of application components secured by the selected user group. The query is similar to the Application Security Query restricted to the current user group.

Copyright © 2024, Oracle and/or its affiliates.

Defining Users

The user maintenance transaction is used to define a user's user groups, data access roles, portal preferences, default values, and To Do roles. To access the user maintenance transaction, select **Admin** > **Security** > **User**.

The user maintenance transaction is the same transaction invoked when the user launches Preferences from the user menu or shortcut key.

Application Security Query

This portal allows a security administrator to review application security configurations across users, user groups and the various types of application components they secure.

While application security is configured at the granular level of an application service and its access modes, it is more intuitive to review access rights to top level broader components:

- **Menu line.** Several application services may be related to a single menu line and impact the ability to properly leverage various aspects of its functionality:
  - The application service of the transaction page or portal, which also controls the visibility of the menu line or its 'search' part when the line is composed of 'add' and 'search' items.
  - The application services associated with any zones on that portal.
  - The application service of the underlying maintenance object when applicable, which also controls the 'add' item of the menu line.
  - The application services of business objects for the underlying maintenance object.
  - The application services of portals used to maintain these business objects.
  - The application services of related portals.
  - The application services of related maintenance objects, their business objects and portals.
  - and more

  You may review the list of application services associated with a menu line on the **Application Security** tab on the **Menu** portal.
- **Sidebar Zone.** This is a regular or context sensitive zone that the user has access to on the sidebar area.
- **System.** This is a Toolbar or other system wide option a user is allowed to use.
- **Batch Control.** This is a batch process a user is allowed to submit.
- **Report Definition.** This is a report a user is allowed to submit.
- **Web Service.** This is a web service a user is allowed to invoke.
- **Miscellaneous.** This is a special service that a user is allowed to invoke.
- **Internal.** This is an internal item that is secured by default but is assumed to be called in a context that does not require security check.
- **Unclassified.** Some application services may not be related to any type of application component and as such remain unclassified. For example, there are situations where previously released services are enhanced to be associated with dedicated application service in a later release. In this situation the services may retain their original or even custom application service and the new application services may appear as not associated with any component.

Refer to Secured Objects and Application Services for details about the types of objects that may be secured.

Open this page using **Admin** > **Security** > **Application Security Query**.

Row Security

Some products allow you to limit a user's access to specific rows. For example, in Oracle Utilities Customer Care and Billing, row level security prevents users without appropriate rights from accessing specific accounts.

Copyright © 2024, Oracle and/or its affiliates.

A combination of framework configuration and configuration in your edge product is required for row level security. The following points describe the configuration:

- For each record that should be secured, associate it with an **Access Group**. Note that if your edge product supports row level security, that product is providing a link between the secure-able record and Access Group. Your access groups may be granular and only referenced by one secured record or they may be more broad and be referenced by multiple secured records that require the same type of security restriction.
- To define which users have access to the secured records, you define a **Data Access Role**. For each data access role, define which Access Groups the role has security clearance for. An access group may be linked to one or more data access roles. In addition, define the **Users** that have access rights to these secured records. When you grant a data access role rights to an access group, you are giving all users in the data access role rights to all secured records in all the referenced access groups. A user may belong to many data access roles.

If your edge product supports row level security, it will include logic in the appropriate areas of the system to limit the secured rows that a user may view or maintain based on this configuration. For example, in Oracle Utilities Customer Care and Billing, throughout the system users are only able to view and maintain information about an account and any of its detail if the user is in a Data Access Role for the account's Access Group (or the account is not linked to an Access Group).

Fastpath: Refer to your product's documentation for more information on row level security, if applicable.

Defining Access Groups

Refer to The Big Picture of Row Security for a description of how access groups are used to restrict access to specific objects.

Access groups control which groups of users (referred to as Data Access Roles) have rights to accounts (or other objects) associated with the access group. You can access the portal using **Admin** > **Security** > **Access Group**. You are brought to a query portal with options for searching for a specific access group. Once a record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The **Access Mode** zone provides the information about the access group and its data access roles.

- The **Data Access Role** collection defines the data access roles whose users have access to the access group's secured objects. Keep in mind that when you add a **Data Access Role** to an **Access Group**, you are granting all users who belong to this role access to all of the objects linked to the access group.

  Note: You can also use Data Access Role - Access Group to maintain a data access role's access groups.
- The **To Do Role Assignment** collection may be used to route To Do Entries of a given To Do Type to a specific To Do Role if the objects related to the To Do Entry are associated with the access group. Refer to Adjusting the To Do Role for more information.

Defining Data Access Roles

Fastpath: Refer to The Big Picture of Row Security for a description of how access groups are use to restrict access to specific objects.

The data access role transaction is used to define two things:

- The users who belong to the data access role.
- The access groups whose accounts (or other objects) may be accessed by these users.

Data Access Role - Main

Select **Admin** > **Security** > **Data Access Role** to define the users who belong to a data access role.

Description of Page

Copyright © 2024, Oracle and/or its affiliates.

Enter a unique **Data Access Role** code and **Description** for the data access role.

The scroll area contains the **Users** who belong to this role. A user's data access roles play a part in determining the accounts (or other objects) whose data they can access.

To add additional users to this data access role, press the add button and specify the following:

- Enter the **User** ID. Keep in mind that when you add a **User** to a **Data Access Role**, you are granting this user access to all of the accounts (or other objects) linked to the data access role's access groups.
- Use **Expiration Date** to define when the user's membership in this data access role expires.

Note: Also maintained on the user page. You can also use User - Access Security to maintain a user's membership in data access roles.

Data Access Role - Access Group

Select **Admin** > **Security** > **Data Access Role** and navigate to the **Access Groups** tab to define the access groups whose accounts (or other objects) may be accessed by the users in this data access role.

Description of Page

Use the **Access Group** collection to define the access groups whose objects can be accessed by this role's users. Keep in mind that when you add an **Access Group** to a **Data Access Role**, you are granting all users who belong to this role access to all of the accounts (or other objects) linked to the access groups.

Note: You can also use Access Group - Main to maintain an access group's data access roles.

Data Privacy

Personally Identifiable Information (PII) is any information that identifies or could be used in combination with other information to identify an individual. Many countries have data privacy laws governing the use of such information. A key aspect of these laws is an organization’s obligation to erase personal information when there is no compelling reason to retain it. This obligation can be fulfilled either by deleting data or altering the data in such a way that it is no longer possible to associate the data with the individual.

Organizations are likely to have other legal obligations that impact how long they retain personal data. For example, there could be financial audit requirements that oblige the organization to retain relevant information for a fixed number of years after termination of a contract.

Object Erasure is designed to address right to erasure issues, with a combination of configuration and processes that may be extended to implement the appropriate policies for the PII within your data.

Encryption and Masking

"Encryption" refers to encrypting data stored in the database using an encryption key. There are two different types of encryption described in the sections below. System encryption refers to columns in the system identified by the product to use encryption. Application encryption refers to the ability for an implementation to configure fields and elements that should be encrypted in the database.

"Masking" refers to overwriting all or part of an un-encrypted field value with a masking character. For example, perhaps only the last 4 digits of a tax ID number are visible with the other digits changed to an asterisk. The system provides support for masking fields on the user interface that may be stored as plain text in the database. In addition, there are cases where encrypted fields are shown to the user interface using masked values rather than the encrypted value.

The following sections provide more information about each feature.

Copyright © 2024, Oracle and/or its affiliates.

System Encryption

The system automatically encrypts certain fields captured in various option tables or context tables. This is mainly used for passwords. For example, passwords captured in Message Sender context or password

In addition, batch control supports configuring a security option for parameters that capture sensitive information, such as a password. Refer to Defining Batch Controls for more information.

It is also possible to enable system encryption using the characteristic type **F1-PWD**. However, the maintenance object must include specific code to enable system encryption for characteristics of this type. In Oracle Utilities Application Framework, the only maintenance object that supports this is extendable lookup. Refer to Extendable Lookup Advanced Topics for more information.

User Interface Masking

The functionality described in this section is used to take data that is stored in plain text in the database and mask the value before it is presented to a user (or an external system). This feature includes the ability to allow some users to view the data unmasked using security configuration. The system allows different masking rules to be applied to different fields. For example, a credit card number can be masked differently than a social security number.

The following topics describe how to mask field values.

Identify the Data to be Masked

Identify the data that is stored as plain text, but should be masked for display to users. For example, imagine that you have identified that Credit Card Numbers and a person's federal ID number (for example, in the United States, the Social Security Number or SSN). Each field identified may be displayed and maintained in different user interfaces throughout the system, but the masking rules for a given field are probably uniform regardless of where the data is displayed.

Primary keys cannot be masked. A field defined as a unique identifier of a row cannot be configured for masking. Masking a field that is part of the primary key causes a problem when attempting to update the record. This restriction also applies to elements that are part of a "list" in an XML column on a maintenance object. One or more elements in the list must be defined as a primary identifier of the list. Be sure that primary key elements in the list are not ones that require masking.

List members that contain different "types". Consider a page with a list that contains a person's identification numbers. You can set up the system so that a person's social security number has different masking rules than their drivers license number. If your implementation has this type of requirement, the list of masked fields should contain an entry for each masking rule.

For each field, if there are some users that may see the data unmasked on one or more of the user interfaces, then security configuration is required. If the value of a field should be masked for all users across all pages in the application, then the security configuration is not needed.

Security Configuration

Define a security type for each field with two authorization levels:

- **1** - Can only see the element masked
- **2** - Can only see the element unmasked

Link all of the security types to an application service of your choosing. We recommend linking every masking-oriented security type to a single application service (e.g., **CM_MASK**) as it makes granting access easier.

For each security type, identify which users can see its data unmasked and which users can only see its data masked. If the masked and unmasked users fit into existing user groups, no additional user groups are necessary. Otherwise, create new user groups for the masked and unmasked users.

After the user groups for each security type are defined, link each user group to the application service defined above. When a user group is linked to the application service, you will define the authorization level for each security type linked to the application service. If a user group's users should see the security type's field values unmasked, set the authorization level to 2; otherwise set it to 1.

Copyright © 2024, Oracle and/or its affiliates.

> **Note:** Flush the cache. Remember that any time you change access rights you should flush the security cache (by entering flushAll.jsp on the URL of the application) if you want the change to take effect immediately.

Configure a Masking Algorithm

A data masking algorithm (using the algorithm entity value of **Feature Configuration - Data Masking**) must be created for each combination of masking rules and security type. These algorithms determine if a user has the rights to view a given field unmasked, and, if not, how the field should be masked.

The base package provides the algorithm type **F1-MASK** whose parameters are designed to handle most masking needs. If certain users may see the data unmasked, parameters capture the application service, security type and authorization level defined above used to evaluate this. In addition, parameters allow you to configure how much of the data to mask, what masking character to use. Refer to the algorithm type description for more information.

Determine How the Fields are Displayed

The masking configuration differs based on how a field is retrieved for access to the user interface. So for the masking of one “logical” field (like a person’s SSN), there may be multiple configuration entries required to cover all the access methods. Review each user interface where a given field is displayed and create the following categories:

- The field is an element that is retrieved by invoking a business object, a business service, or a service script
- The field is displayed on a fixed maintenance page (and is therefore retrieved by invoking a page service)
- The field is displayed on a fixed search page (and is therefore retrieved by invoking a search service)
- The field is stored as an ad hoc characteristic

Create a Feature Configuration for Each Masked Element

Create a feature configuration with a Feature Type of **Data Masking**. An option entry with option type of **Field Masking** is needed for every combination of field to mask and the method used to display the data. The value will contain mnemonics that reference the appropriate data masking algorithm along with configuration that differs depending on how the field is retrieved for display as described below.

Schema Based Object Field Masking

For data that is accessed via a schema-based object call and displayed in a UI map, the field to be masked must reference a meta-data field name in its schema definition: **field="fld_name", alg="algorithm name"**

If the element references an mdField in the schema, that is the field used to identify the masking rule. If there is no mdField reference but only a mapField reference, that is the field used to identify the masking rule. For example, if you want to mask a credit card number, let's assume that field is defined in the schema is the following:

```
<creditCard mdField="CCNBR" mapField="EXT_ACCT_ID"/>
```

In this case, the option value should be **field="CCNBR", alg="algorithm name"**. An option value of **field="EXT_ACCT_ID", alg="algorithm name"** would not result in masking.

A "where" clause may also be specified. This is useful for data that resides in a list where only data of a certain type needs to be masked: **field="fld_name", alg="algorithm name", where="fld_name='value'"**

For example, person can have a collection of IDs and only IDs of type 'SSN' (social security number) should be masked. If the person data including its collection of person IDs is displayed on a UI map via a business object call, let's assume the collection is defined in the following way:

```
<personIDs type="list" mapChild=CI_PER_ID">
  <isPrimaryId mapField="PRIM_SW"/>
  <idType mapField="ID_TYPE_CD"/>
  <personIdNumber mapField="PER_ID_NBR"/>
</personIds>
```

Copyright © 2024, Oracle and/or its affiliates.

The option value may look like this: **field="PER_ID_NBR", alg="algorithm name", where="ID_TYPE_CD='SSN'"**

Please note the following important points for schema based masking:

- **Limitation of 'where' field** Although the main use of a 'where' clause for schema oriented elements is to mask certain elements in a list based on a 'type', it is also possible to mask a single field in the schema based on the value of another field. For example, imagine that a customer submits a registration form that defines an ID type and ID value. Although this data is not in a list, the implementation may still want to only mask the ID value if the ID type is "SSN". The framework is only able to mask an element in the schema based on a 'where' clause if the element in the 'where' clause is a "sibling" in the schema.
  - If the element to be masked is in a list, the element in the 'where' clause must be in the same list.
  - If an element to be masked maps to a real column in a table, the element in the 'where' clause must also map to a real column in the table.
  - If an element to be masked maps to and XML column in the table as a single element, the element in the 'where' clause must map to the same XML column as a single element.
- **Multiple feature option entries for the same field.** It's possible that different schemas in the system have a similar type of data that may be masked based on different conditions. For example, imagine that an implementation has different schemas that captured or referenced person identifiers in different ways:
  - One schema captures a single person ID without any corresponding "type" record and it should always be masked using Algorithm CM_SSN_MASK:

    ```
    <personSSN mapXML=BO_DATA_AREA mdField=PER_ID_NBR/>
    ```

  - One schema captures a person ID and a corresponding ID Type and it should be masked with Algorithm CM_SSN_MASK if the type is "SSN" and masked with algorithm CM_FEIN_MASK if the type is "FEIN".

    ```
    <personIdType mapXML=BO_DATA_AREA mdField=ID_TYPE_CD/>
    <personId mapXML=BO_DATA_AREA mdField=PER_ID_NBR/>
    ```

  - One schema captures a person ID and a corresponding ID Type and it has the same masking rules as the previous schema, but a different field name is used for the ID Type code. (This scenario could happen if for example a different label is desired for ID Type on the user interface for this schema.)

    ```
    <personIdType mapXML=BO_DATA_AREA mdField=CM_ID_TYPE/>
    <personId mapXML=BO_DATA_AREA mdField=PER_ID_NBR/>
    ```

  For this scenario, the feature options may look like this:

  1. **field="PER_ID_NBR", alg="CM_SSN_MASK"**
  2. **field="PER_ID_NBR", alg="CM_SSN_MASK", where="ID_TYPE_CD='SSN'"**
  3. **field="PER_ID_NBR", alg="CM_FEIN_MASK", where="ID_TYPE_CD='FEIN'"**
  4. **field="PER_ID_NBR", alg="CM_SSN_MASK", where="CM_ID_TYPE='SSN'"**
  5. **field="PER_ID_NBR", alg="CM_FEIN_MASK", where="CM_ID_TYPE='FEIN'"**

  For each schema, the system will first find whether the element applies to any masking option. It will find 5 masking options for the field PER_ID_NBR. Then it will determine if any sibling elements match the 'where' clause.

  - If more than one sibling element matches a 'where' clause, a runtime error is issued. For example if a schema has an element that references "mdField=ID_TYPE_CD" and an element that references "mdField=CM_ID_TYPE", this is an error. Additionally, if multiple elements reference mdField=ID_TYPE_CD", this is an error.
  - If one and only one sibling element matches a 'where' clause, the value of the element is compared to the values defined in the 'where' clause. If it finds a match on the value, the appropriate masking algorithm is applied. If no match is found (for example, the Person ID Type is "LICENSE") the element is displayed as is.
  - If no sibling element matches a 'where' clause and a feature option exists with no 'where' clause (option 1 above), then the masking algorithm of the option with no 'where' clause is applied.
- **Changing the value in the 'where' clause.** If your implementation has some users that are allowed to change records where some data is masked based on a condition, it is recommended to design the user interface to reset the masked

Copyright © 2024, Oracle and/or its affiliates.

value when the value in the 'where' clause changes. For example, if a user is prevented from viewing a person's social security number, but the user is allowed to make updates to the person's record, changing the value of the Person ID Type should reset the Person ID Number. This would ensure that the user does not 'unmask' the social security number by simply changing the ID Type.

Records Maintained Using Page Maintenance

For data that is accessed via a page maintenance service call, indicate the table name and the field name where the data resides: **table="table_name", field="fld_name", alg="algorithm name"**

For example if the Person record and its collection of identifiers are displayed and maintained using page maintenance, the option value should be **table="CI_PER_ID", field="PER_ID_NBR", alg="algorithm name"**

A "where" clause may also be specified: **table="table_name", field="fld_name", where="fld_name='value'", alg="algorithm name"**

This is useful for data that resides in a child table where only data of a certain type needs to be masked. For the person ID example, **table="CI_PER_ID", field="PER_ID_NBR", alg="algorithm name", where="ID_TYPE_CD='SSN'"**

Characteristic Data

For data that is stored as a characteristic, simply indicate the characteristic type: **CHAR_TYPE_CD='char type', alg="algorithm name"**

This needs to be defined only once regardless of which characteristic entity the char type may reside on. Note that only ad-hoc characteristics are supported.

Masking Fields in Explorer Zones or Info Strings

In explorer zones data is often retrieved using SQL directly from the database. No masking is applied automatically in this case. If there is data in the explorer zone results that should be masked, the masking must be applied by calling a business service.

Similarly, an MO Info algorithm may not use BO interaction to get data. It may access data using SQL for efficiency purposes. No masking in applied when retrieving data via SQL. To apply masking to a string prior to including it in an info string, the masking must be applied by calling a business service.

The system supplies two business services to be called to determine if masking rules apply for a specific field.

- **F1-TableFieldMask**. Mask a Table field. This business service receives a table name, field name and one or more field values. If masking applies it returns the masked value.
- **F1-SchemaFieldMask**. Mask a Schema field. This business service receives a schema name and type, XPath and field value. If masking applies it returns the masked value.

Search Service Results

For data that is displayed on a 'fixed' search page, it is retrieved via a search service call. Indicate the search name and the appropriate field to mask along with the masking algorithm. For example: **search="SearchServiceName", field="PER_ID_NBR", where="ID_TYPE_CD='SSN'", alg="algorithm name"**

To find the name of the search service, launch the search in question, right click in the filter area and choose **Inspect**. Search for "serviceName". The service name is listed there. To find the field name to mask, search for "Widget Info". There should be two results found, one for the filter area and one for the results area. This results area has the text "SEARCH_RESULTS" as prefix for each field. The field names are the ones after the x$. Do not reference the x$ when defining the field name. Note, the "where" statement can only apply to fields that are also part of the search results.

Additional Masking Information

The following points provide additional information to assist in your masking configuration:

- If the demonstration database includes a **Data Masking** feature configuration, review the settings because it will probably contain masking rules that will match your own.
- On data input pages, a user might be able to enter or change masked data, such as a bank account number, but not be able to subsequently see what they added or changed.

Copyright © 2024, Oracle and/or its affiliates.

- External systems can request information by performing a service call via a web service. Please keep in mind that some web service requests require data to be masked and some do not. For example, a request from an external system to synchronize person information needs the person's social security number unmasked; whereas a request from a web self service application to retrieve the same person information for display purposes needs the person's social security number masked. To implement this type of requirement, different users must be associated with each of the requests and these users must belong to separate user groups with different access rights.
- If a maintenance object (MO) contains a field that holds an XML document and a service call invokes the MO's service program directly, the system will mask individual XML elements in the field if a **Determine BO** algorithm has been plugged into the maintenance object and the element(s) in the respective BO schema have been secured as described above.

Application Encryption

The functionality described in this section allows implementations to configure fields to encrypt when storing it in the database. This functionality is mutually exclusive from the User Interface Masking functionality described in the previous section. This feature supports encrypting specific elements stored within a CLOB or XML column.

The following points highlight the features of the encryption functionality:

- The encryption key is defined using a keystore, which must be set up in order to use this functionality. For details about setting up the keystore in the system, see the Installation Guide.
- When a field is configured to be encrypted, the encrypted data is stored in a special encryption field that is not the source field (the one exposed to the user on the user interface). The source field captures the data as masked. Because a special field is required to support encryption, the product must provide support for that field to be encrypted.
- For encrypted data that must allow searching, the system supports capturing a hash value in a special field. The product must provide support for this functionality. Besides providing a special field to capture the hash value, base search functionality for that data must also cater for this configuration.
- The system supports encrypting data that is captured as an element within an XML field. If the XML field is provided in a schema owned by the product, then the product must provide specific support for the capture of the encrypted data.

The following sections provide additional information about the support for encryption provided by the framework. Refer to the security chapter of the administration guide for your particular product for more information.

Encrypting and Masking the Data

When a product enables encrypting for a given type of data, a special encryption field should be created to capture the encrypted value. Because encrypting is optional, the source field (the one exposed to the user) should not be this special encrypted field. If encryption is configured, the system will internally populate the encrypted field. The source field will be populated with asterisks by default. That way the masked data is what is shown to the user on page rather than the encrypted value.

The following points highlight how the system behaves when encryption is configured and when it is not. Assume as an example, the field is a credit card number. The user views and populates a field with the field name CC_NBR. The table also has a second field ENCR_CC_NBR. A user populates the credit card number:

- If encryption is not configured, CC_NBR will be updated with the entered credit card number and ENCR_CC_NBR will be empty. Note that in this case, an implementation may choose to configure user interface masking.
- If encryption is configured, CC_NBR will be updated with '**********' and ENCR_CC_NBR will contain the encrypted value. The asterisks for the standard field will fill the full field size up to 50 characters.

If for some reason the standard masking using all asterisks is not desired, the system supports supplying an explicit masking algorithm using the same Feature Configuration - Data Masking plug-in spot used for User Interface Masking.

Warning: Unlike user interface masking, the masking of encrypted fields is not driven by security. The data stored in the source field for all encrypted data should be masked. Be sure not to configure security authorization logic in algorithms used for this type of masking.

Copyright © 2024, Oracle and/or its affiliates.

Feature Option Configuration

Create a feature configuration with a Feature Type of **Encryption**. For each source field you are encrypting, enter an option with option type of **Field Encryption**. The value will contain mnemonics that reference the appropriate encryption key alias defined in the keystore along with configuration related to the field and its table location. Unlike the user interface data masking, the configuration for data encryption is related to how the data is stored rather than how it is displayed. In addition, each entry may define an explicit masking algorithm to override the default and if supported, may also define a hash field and hash alias.

For data that is stored in a specific column on a table, an explicit field to capture the encrypted value must exist. Indicate the table name, source field name and encrypted field name along with the alias: **table='table_name', field='fld_name', encryptedField='encr_fld_name', alias='alias key'**

A "where" clause may also be specified when data resides in a child table and only data of a certain type needs to be encrypted.

Example, **table='CI_PER_ID', field='PER_ID_NBR', encryptedField='ENCR_PER_ID_NBR', alias='key alias', where='ID_TYPE_CD='SSN''**

For data that is stored in an XML column in a record, the source field to be encrypted must reference a meta-data field name in its schema definition along with the element that captures the encrypted data and the alias: **field='field_name', encryptedField='encr_field_name', alias='key alias'**

The syntax for adding a reference to a masking algorithm is **maskAlg='algorithm name'** .

The syntax for adding configuration for capturing a hash value for searching purposes is **hashAlias='hashAliasKey' hashField='HASH_FLD_NAME'**.

The following is an example of configuration that uses all the possible options (specific masking algorithm, where clause and hash field support):

table='CI_PER_ID', field='PER_ID_NBR', alias='aliasKey', encryptedField='ENCR_PER_ID_NBR', hashAlias='hashAliasKey' hashField='HASH_PER_ID_NBR', where='ID_TYPE_CD=SSN', maskAlg='CM-PERIDMASK'

Searching by an Encrypted Value

If the product supports a hashed value for an encrypted field for searching purposes, the following points highlight explorer zone configuration for this purpose

- The user filter value should reference the source field and should include an additional **encrypt=** mnemonic. For example

  ```
  type=STRING
  label=PER_ID_NBR -
  encrypt=[CI_PER_ID,PER_ID_NBR,ID_TYPE_CD,F1]
  ```

  Refer to User Filters for more information.
- The SQL should include the hashed value in the WHERE clause. Note that because encryption is optional, a product zone that includes searching by a field eligible for encryption will include finding a match for the filter in the source field (as plain text) or in the hashed field. For example:

  ```
  WHERE
     [(F2) (ID.PER_ID_NBR =:F2 OR ID.HASH_PER_ID_NBR = -:F2)]
  ```

Customizing Encryption Algorithm

Although the encryption algorithm to use with a given key can be gleaned from the key in the keystore, there is sometimes extra information associated with an algorithm that might need to be used to encrypt or decrypt data.

The system provides a feature configuration option for the **Encryption** feature type using the option type **Algorithm Info** that can be used to adjust the behavior of the encryption.

Copyright © 2024, Oracle and/or its affiliates.

- You can modify the default mode and padding of the encryption algorithm.
- If a key will be used to digitally sign anything, the signing algorithm can also be specified for the key.

For details about the syntax, refer to the feature option type's detailed description.

The Approach to Implementing Object Erasure

This section describes the product approach to implementing object erasure for its maintenance objects (MOs).

Note: The approach outlined is intended for managing erasure of master data objects. The assumption is that Information Lifecycle Management will be used for archiving any related transactional data.

Various events in the system may signal the need to erase data at some future date. For example, closing an account may indicate the need to erase both the account and person data once there is no longer a need to maintain related financial details. There may also be events that re-establish the business relationship with a person and signal that data marked for erasure should now be retained.

The Object Erasure Schedule is used to capture key details of objects with data that needs to be erased. Records in the schedule are monitored periodically to determine if the date for erasure has been reached and if so, execute the erasure logic. The system will create and maintain records in the schedule when specific events occur for an MO that is marked for erasure.

The system provides the ability to mark an MO as eligible for erasure via a number of MO configuration entries. Included in these is an option that defines the business object for the MO's erasure schedule records. In addition to defining the erasure record elements, this BO defines the monitoring process and the specific erasure logic for the MO.

The following sections provide more information about configuring and managing object erasure.

Configuring a Maintenance Object for Erasure

There are three types of maintenance object configuration entries that define how to manage erasure:

- An option to define the business object for erasure schedule records.
- An option to define the period between the date on which the need for erasure is detected and the date on which erasure should occur. The common logic provided by the system to add or update erasure records uses this option to determine the erasure date.
- One or more entries in the MO algorithm collection for the algorithms that determine whether a record should be scheduled for erasure, according to the current state of the record, and create or update the erasure schedule entries.

Some MOs that would commonly be regarded as eligible for erasure will be installed with a base Erasure BO already configured. The expectation is that the corresponding Erasure Period option and the algorithm for the Manage Erasure Schedule plug-in spot will be set up by your implementation with the appropriate entries for your organization's business rules.

Note: The presence of an Erasure Period option is the signal to the system that your implementation has enabled an object for erasure.

It may not be necessary for all master data objects that contain personally identifiable information to manage their own erasure schedule. For example, the event which triggers erasure for a person record may also trigger the creation of erasure schedule entries for the person's other master data records. In this case, only the person maintenance object will need to be configured with a Manage Erasure algorithm (or algorithms). Refer to your product specific documentation for more information on the recommended approaches for your product's master data objects.

Manage Erasure Schedule Algorithm

Algorithms of this type are responsible for determining a record's erasure status and creating or maintaining an entry in the Object Erasure Schedule for the record. They are triggered when certain events occur in the system for maintenance objects that are eligible for erasure. Multiple algorithms can be configured for the plug-in spot.

Copyright © 2024, Oracle and/or its affiliates.

Depending on the determination made by the algorithm (or algorithms), a number of different actions may need to be taken. For example:

- A new schedule record may need to be added
- An existing schedule record may need to be updated with a new erasure date
- A pending record may need to be discarded if erasure is no longer applicable
- A discarded record may need to be reactivated if erasure now applies again

The base business service **F1-ManageErasureSchedule** performs the common logic to handle the possible actions and is recommended for use by these types of algorithms. The service can also optionally create a log entry for information purposes. Refer to the description of the business service for full details.

In certain circumstances, if one record is scheduled for erasure, other related records may need to be evaluated. The base business service **F1-ManageErasureScheduleDriver** performs the logic to execute an MO's Manage Erasure Schedule algorithms. It can be used to manage any related records from within another algorithm.

This type of algorithm is plugged into the Maintenance Object - Algorithm collection.

Your product may supply algorithms to manage the schedule for base MOs that are regarded as eligible for erasure. If your organization has special business rules that are evaluated to determine the erasure schedule for an MO, a custom algorithm can be created and applied by the implementation team.

Monitoring the Schedule and Performing Erasure

The erasure schedule is managed using the lifecycle of the erasure record's business object. A deferred monitor process is used to periodically check for records that are due for erasure and transition them from the 'pending erasure' state to the 'erased' state. An enter plug in on the 'erased' state is responsible for performing the erasure logic.

The system provides a 'root' business object for the object erasure schedule (**F1-ErasureScheduleRoot**) which defines the lifecycle that erasure schedule business objects should follow. The system also provides a monitor process (**F1-OESMN**) which is configured on the 'pending erasure' state of the 'root' business object. This process is configured to monitor object erasure schedule records whose erasure date is on or before the batch business date.

Maintenance objects that are eligible for erasure should be configured with an erasure schedule business object that uses the 'root' business object as its parent. The algorithm that performs the erasure processing applicable to that maintenance object must be plugged in on the 'erased' state of the 'child' business object.

The system provides an erasure schedule business object (**F1-ErasureScheduleCommon**) which is designed to erase an object by deleting the main record and any child records. Your product may supply additional erasure schedule business objects for certain use cases. Refer to your product specific documentation for more information.

Erasing User Information By Obfuscation

Some implementations consider the base User object to have personally identifiable information. Erasing user information by deleting the records is not advisable as it can cause referential integrity problems. The recommended approach is to obfuscate the data instead.

The system provides a user erasure business object (**F1-ErasureScheduleUser**) which is designed to support obfuscating a user's identifiable information by removing it or replacing it with a non-identifiable value if the field is required.

If the User maintenance object is configured to be eligible for erasure in your implementation, the manage erasure schedule algorithm for the record will be invoked whenever the **Enable** flag is changed for a record on the User maintenance page. The system provides a base algorithm (**F1-OBJERSUSR**) specifically for the User MO. This algorithm will add an entry to the object erasure schedule if the user record is disabled or deactivate an existing erasure schedule record if a disabled user record is enabled again.

Viewing an Object's Erasure Status

The system provides a portal for viewing and editing an erasure schedule record. The portal is accessed from the **Object Erasure Schedule** sidebar zone. This is a context sensitive zone which only appears when accessing a record for a

Copyright © 2024, Oracle and/or its affiliates.

maintenance object that is configured to be eligible for erasure. It displays the information for the erasure schedule entry for the record in context, if applicable.

To add the zone to an eligible maintenance object:

- Navigate to Context Sensitive Zone and search for the navigation key for the maintenance object.
- Add the Object Erasure Schedule zone **F1-OBJESRSCD** to that navigation key.

Viewing Erasure Configuration

The Erasure Configuration All-in-One portal provides the ability to view the maintenance objects that are marked as eligible for erasure and maintain their erasure configuration entries in a single place.

To view a list of the eligible MOs, open **Admin** > **Security** > **Erasure Configuration.**

Press the edit icon to open a window in which the configuration values for that row can be changed.

Refer to The Approach to Implementing Object Erasure for an overview of object erasure configuration.

Archiving the Object Erasure Schedule

The system provides the ability to archive older records in the object erasure schedule using Information Lifecycle Management. The object erasure schedule maintenance object is configured with a base ILM eligibility algorithm and an ILM crawler batch control.

The base object erasure schedule BO is configured to set the ILM archive switch when an erasure schedule record enters a final state and to reset it if the record becomes active again.

Refer to The Approach to Implementing Information Lifecycle Management for more information on archiving functionality.

Data Redaction on Export

Personal identifiable information (PII) requires special handling when exported out of the application and written to files. Information that is encrypted at the database level remain encrypted and therefore protected when exported. With respect to non-encrypted information, the need to obfuscate it for added protection depends on how the exported data is intended to be used. If the exported information is part of an integration flow with an external system then the data should remain unchanged. However, when exporting data from a production environment for testing purposes sensitive information should be anonymized before they can be imported to a lower environment.

The following are key aspects of this functionality:

- Redaction rules define how specific fields should be redacted when exported.
- When redaction rules exist, data redaction is performed by default by specific export tools. In absence of redaction rules, data is exported in its original form.
- Redaction rules are customer specific and may only be relevant to environments that contain sensitive data. As such, they are not released with the base product. Review your data redaction requirements and configure appropriate rules in relevant environments.
- Certain users may be allowed to export data in a non-redacted form based on application security.

The following sections provide additional information about this topic.

Redaction Rules Are Customer Specific

Redaction rules identify which fields should be redacted and how they should be anonymized. The list of fields to redact and the function to use to manipulate their values are customer specific and therefore are not released with the base product. A sample set of rules are provided as accelerator data for reference only.

It is important that you review your data redaction requirements and configure rules that reflect your business needs.

Copyright © 2024, Oracle and/or its affiliates.

Redaction By Default

Processes that support data redaction on export do so by default when redaction rules exists. When rules are not defined, data is exported as is. Note that for this reason it is unnecessary to define such rules in lower environments that do not contain sensitive information.

Specific Export Tools Only

Data redaction is supported by the following export tools only:

- **Content Migration Assistant (CMA).** Depending on application security, a user may indicate on a specific data set export request that data should be exported in its original form.
- **General Data Export (GDE) batch processes.** A batch parameter may be used to explicitly request that data is exported in its original form. This option is also protected by application security and enforced at batch submission time.

> **Note:** This type of process establishes a single export stream that may be sent to multiple targets. Review these integrations to ensure they do not have conflicting data redaction requirements and set the batch parameter accordingly.

Different Types of Redaction Rules

Sensitive information may reside in various types of storage fields which impact the way they are identified for redaction. You may define rules that target fields on specific tables or provide a more global rules that applies to all references of a specific field.

The application supports the following types of reduction rules:

- **Physical Field.** This type of rule applies to a value residing in a field on a specific table. An optional filter expression may be provided when only records of a certain type needs to be redacted.
- **XML Storage.** This type of rule applies a value associated with the specified field name that resides within an XML storage field on a specific table. This option also supports a filter expression.
- **Reference Field.** This type of rule applies to a value residing anywhere and associated with the specified field name. It does not include fields referenced in XML storage fields.

Redaction Function

A redaction function defines the format by which a value is anonymized. The type of function used depends on the field's data type. For example, to anonymize a numeric field the redaction function should only use numeric values etc.

Refer to the **Redaction Function** extendable lookup for a list of available functions. The base product provides a sample of common functions but the list may be extended by your organization as needed.

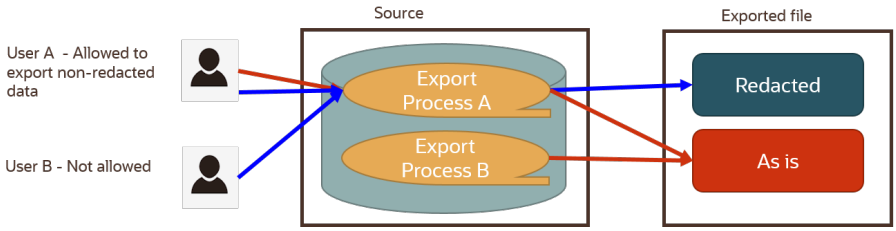
Fields May Contain Sensitive Information

The field definition record includes an indication as to whether the field may contained sensitive information or not. Note that the indication does not imply that the field is actually used to capture such data but rather that it can potentially be used this way. Use this indication to review fields that may need to be associated with a redaction rule.

Exporting Non-Redacted Data Requires Security Rights

By default, the export tools redact data based on existing redaction rules. There are situations where certain users may be allowed to export data in its original form, that is, to request to not apply the redaction rules.

Copyright © 2024, Oracle and/or its affiliates.



The following points describe the processes for exporting data:

- Some processes always export data as is, in case the target consumer needs the raw data. This is irrespective of the user requesting the export.
- Some processes perform data redaction by default and may offer an option for exporting data as is. The option to export data in its original form is secured by application security.
- In this example, process A supports data redaction by default with an option to export non-redacted data. User A has the right to export the data redacted (default) or explicitly request to export it in its original form. User B does not have the right to do so hence their export content is redacted by default.

The option to export data in its original form is associated with the **Do Not Apply Redaction Rules** access mode associated with the following relevant application services:

- For CMA, the additional access mode is associated with the migration data set export's application service. When granted, the user may select the special non-redacted export option when they create the export data set request.
- For GDE batch processes, the access mode is associated with their application service and is tied to the **Do Not Apply Redaction Rules** batch parameter they support. Only users with the proper application security can submit these batches using the special batch parameter option

Defining Redaction Rules

This portal is used to view and maintain rules that define how specific fields that containing personal information should be redacted when exported out of the application. For example, an e-mail address may be redacted by replacing some or all characters with '*'.

Refer to Data Redaction on Export for more information.

You can access the portal from the**Admin Menu** > **Security** > **Redaction Rule**. You are brought to a query portal with options for searching for a specific redaction rule. Once a redaction rule has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page

- **Redaction Rule.** This zone provides information about the selected rule.

Obfuscation Configuration

This portal is used to view and maintain all configuration options related to personal information obfuscation. It includes all masking, encryption and data redaction rules.

You can access the portal from the**Admin Menu** > **Security** > **Obfuscation Configuration**.

Refer to Encryption and Masking for more information.

The following zones may appear as part of the portal's **Main** tab page

- **Redaction Rules.** This zone lists all redaction rules. You may add a new rule from this zone.
- **Masking.** This zone lists masking options. You may navigate to the masking related feature configuration to review and edit these options.
- **Encryption.** This zone lists encryption options. You may navigate to the encryption related feature configuration to review and edit these options.

Copyright © 2024, Oracle and/or its affiliates.

- **Review Fields for Redaction Rules.** This zone lists fields that may contain personal identifiable information and are not associated with a redaction rule. Review these fields and add redaction rules when applicable.

Cryptography Keys

Often when communicating information with an external system, cryptography keys are used to exchange encrypted information or confirm that the two communicating parties recognize each other and the information being provided.

The following sections include information about the functionality provided to support this type of security.

Understanding Key Rings

Cryptography keys may be used to provide a signature or credentials to a request so that the system recognizes that the request comes from a trusted party. Keys may also be used to encrypt or decrypt files shared between two parties.

The Key Ring object is provided to reference the keys that are used over time for a given business use case. Only one key or key pair may be active at any given time.

The following sections include information about the functionality provided to support different key ring classes for particular use cases.

RSA Signature Keys

RSA signature key pairs are used in the following cases:

- Certain processes in the system may need to write files to or read files from Oracle Cloud Object Storage. Cloud object storage has its own method for encrypting files, and products using the framework application do not need to define additional configuration for that. However, when communicating with cloud object storage, the application is required to provide a signature key.

  Refer to External File Storage for more information about configuring the system to connect to cloud object storage.
- Customers may need the ability to authorize a customer service representative (CSR) to view the customer's account information in a customer facing portal. A signature key is required to support secure communication with the endpoint.

  Refer to User Redirection for more information about configuring the system to enable this type of redirection.

The product provides an RSA Signature Key Pair business object that supports the following functions:

- Generating a key pair, public and private. The private key is stored in an appropriate "secret store" and the public key is available to copy and share with a third party.
- Viewing the public key, allowing a user to copy it to register it with cloud object storage.
- Activating the new key pair (to be done after registering the new public key).
- Inactivating an expired key.
- Key rotation. For increased security, a new key pair should be generated periodically.
- Deactivating all keys for the key ring.

OAuth Keys

The system provides support for sending messages secured by OAuth authentication. To deliver the message, the application is required to provide a client ID and secret. The OAuth secret is sensitive data and needs to be stored securely in the application. The Message Sender configuration provides the ability to store the client ID and an encrypted secret directly in the sender context. Alternatively, the sender context can refer to a key ring that stores those details. Using a key ring allows users to apply new secrets over time without having to update the affected message sender configurations.

Refer to Message Sender - Context for more information about configuring a message sender for OAuth security.

The product provides an OAuth Keys business object that supports the following functions:

Copyright © 2024, Oracle and/or its affiliates.

- Defining the external system related to the key ring
- Adding a client ID and secret key. The secret key is stored in an appropriate "secret store".
- Key rotation. For increased security, a new secret key may be entered periodically. The new key automatically becomes the active key.

File Encryption and Signing Keys

For security, files exchanged with third parties may be encrypted, digitally signed or both. This requires the sender and recipient to maintain public/private key pairs and share public keys. The product supports the maintenance and generation of public and private keys in a format suitable for encryption using PGP standards.

The following sections provide additional information about encryption keys.

Encryption Key Pairs

Internal encryption key pairs are used in the following cases:

- Creating a digital signature for an extract file. The signature is encrypted with the private key of the key pair and decrypted by the recipient using the shared public key .
- Decrypting an import file. The expectation is that the sender has encrypted the file data using a temporary key. The temporary key is encrypted using the shared public key and decrypted using the private key.

The product provides an Encryption Key Pair business object that supports the following functions:

- Generating a key pair, public and private. The private key is stored in an appropriate "secret store" and the public key is available to copy and share with a third party. Both keys are in formats suitable for PGP encryption.
- Viewing the public key, allowing a user to copy it to register it with third parties.
- Activating the new key pair (to be done after sharing the new public key).
- Inactivating an expired key.
- Key rotation. For increased security, a new key pair should be generated periodically.
- Deactivating all keys for the key ring.

Public Encryption Keys

External public encryption keys are used in the following cases:

- Verifying the signature of an imported file. The signature is decrypted using the public key provided by the sender.
- Encrypting an extract file. The batch file adapters create a temporary key which is used to encrypt the file data. The temporary key is encrypted using the public key provided by the recipient

The product provides a Public Encryption Key business object that supports the following functions:

- Defining the external system and reference ID related to the key ring.
- Adding a public key to the key ring. The product expects the public key to be in a format suitable for PGP encryption.
- Viewing the public key.
- Inactivating an expired key.
- Key rotation. When new public keys are provided by the external system, the previously active key is set to expiring status. It may be manually inactivated after all messages using that key have been processed.
- Deactivating all keys for the key ring.

Defining Key Rings

Refer to Understanding Key Rings for an overview of key ring functionality.

To maintain the key rings applicable to your product or implementation, open **Admin** > **Security** > **Key Ring.**

This is an All-in-One portal and includes the standard List and display zones for a key ring.

Copyright © 2024, Oracle and/or its affiliates.

The information captured on the key ring depends on the business objects supported by your product or implementation. Refer to the embedded help text for more information.

Key Zones

The key zones provide information about the keys that are associated with the key ring. The zones displayed and the available actions depend on the type of key. Refer to embedded help for more information.

Key Rotation

For internal keys, it is recommended to practice key rotation and generate new keys periodically. New key pairs are generated in the **Pending** state. The user should register the public key of the new key pair with the associated third party. Once that is done, the new key should be activated.

For external keys, new public keys are added in the **Active** state. The previously active key is set to **Expiring** and may be inactivated when no further processing needs to refer to it.

Copyright © 2024, Oracle and/or its affiliates.

Chapter 3

User Interface Tools

Topics:

- Understanding Portals and Zones
- Setting Up Portals and Zones
- Defining Menu Options
- Defining Navigation Keys
- Defining Navigation Options
- Understanding System Messages
- Understanding Unified Search
- Trees
- Contextual Insights
- Color Configuration
- Defining Display Icons
- Custom Look and Feel Options

This section describes tools that impact many aspects of the user interface.

Understanding Portals and Zones

A portal is a page that is comprised of one or more information zones. The base product pages are built using either a fixed page metaphor or using portals and zones.

There are three broad classes of portals:

- **Standalone.** Standalone portals are separate pages where the main tab of the page is built using a portal. These pages are opened using any of the standard methods (e.g., by selecting a menu item, by selecting a favorite link, etc.). Additional tabs for a stand-alone portal may be included using tab page portals.
- **Tab Page.** These types of portals cannot be attached to a menu. They simply define the zones for a tab on a standalone portal.
- **Sidebar.** The sidebar portal is a special portal that appears in the Sidebar Area on the user's desktop. Its zones contain tools and information that exist on the user's desktop regardless of the transaction. There is only one sidebar portal. This portal and several zones are delivered as part of the base-package. Your implementation can add additional zones to this portal. Please contact customer support if you need to add zones to the sidebar portal.

The contents of this section describe general information about portals and zones.

Portal Page Structure

A portal page has a main tab and may have additional tabs. The page is constructed by first creating a **Standalone** portal, which is also the main portal, followed by creating a **Tab Page** portal for each additional tab and linking it to the main portal. Content is organized on each tab using independent zones.

The entire page is secured by the single application service associated with the main portal.

Portals Are Made Up of Zones

A portal is a page that contains one or more zones, and each zone contains data of some sort. All zones reference a **Zone Type**. The zone type controls the behavior of the zone and the parameters available to configure the zone.

Not All Zones Make Up Portals

Zones may or may not be associated with a portal. The following describes additional uses of zones:

- A data explorer type of zone may implement a search for an entity on an input user interface map. These zones may be referenced on a FK reference record or directly referenced on a UI map.
- A data explorer type of zone may implement a Business Service. The latter may be used for an internal process or used to build a user interface dropdown.
- A zone may be defined as a **Context Sensitive Zone** that appears on the Sidebar when a specific page or portal is in context. This type of zone is not linked to the Sidebar portal but associated with the context page or portal's navigation key.

Configuring Zones for a Portal

The portal includes configuration of how the zones should appear on the portal by default. This includes the following options, all of which may be overridden by an implementation.

- The order in which the zone should appear. An implementation may configure an override sequence to change the order zones on a base delivered portal.
- Whether the zone is visible on the portal. Zones delivered in the base product should be configured to be visible. But an implementation may override this if desired.
- Whether the zone should display initially collapsed or not. A zone's data is only retrieved when it is expanded. As such, a zone may be configured to be initially collapsed when the data is not needed very often. A user can expand the zone when the information is required. Implementations may change the collapsed setting of a base product portal / zone. Refer to Zones May Appear Initially Collapsed When a Page Opens for more information.

Copyright © 2024, Oracle and/or its affiliates.

Flexible Layout Control

By default a zone spans the full width of the portal's display area but can be configured in a number of ways using the layout control features on the portal.

These features allow portal designers to have finer-grained control over how zones are laid out, and enables them to achieve a better user experience and a more optimal layout that reduces unused space.

The following highlight key layout control options:

- In addition to the standard half and full width options, 1/4, 3/4, 1/3 and 2/3 width options are also available.
- Better handling of zones that are dynamically hidden by a visibility script. When a zone becomes invisible its space may be reclaimed by a next zone as long as it is of same width or smaller.
- A zone may define the beginning of a new row. This indication allows for a more fixed layout that is not impacted by hidden zones. The new row indication may establish a layout of one, two or three zones displayed vertically.
- Support the following zone height options:
  - **Content (Unlimited Height)** lets the height be determined by content.
  - **Content (Limited Height)** lets the height be determined by content up to a specified maximum limit. This option may be used to achieve a more consistent portal layout, especially when a zone is rendered on long or small devices.
  - **Fixed Height** sets the zone at a specific height regardless of content.

    This option too may be used to achieve a more consistent portal layout.
  - When a height option is not explicitly selected, zone height is unlimited and determined by content. However, for a data explorer zone, the results grid area itself (not the entire zone) is limited by the zone parameter "height of report" (which has a default value of 50%).

User Preferences

A **Standalone** portal includes configuration to indicate whether or not the portal should appear on a user's portal preferences. The portal will not appear on the user's portal preferences if the **Show on Portal Preferences** setting is set to No. Note that an implementation may change this value for a product delivered portal. This setting is typically enabled for a portal that provides disparate information where not all zones are applicable to all users or where users may wish to adjust the order of the zones. An example of a portal enabled for portal preferences is the Sidebar portal.

The user can override zone oriented configuration for the portal:

- Which zones appear on that portal
- The order in which the zones appear
- Whether the zones should be initially collapsed when the portal opens.
- The refresh seconds. This is applicable to zones displaying data that changes often.

An implementation can optionally configure the system to define portal preferences on one or more "template" users. When a template user is linked to a "real" user, the real user's preferences are inherited from the "template" user and the "real" user cannot change their preferences. Some implementations opt to work this way to enforce a standard look and feel for users in the same business area.

Refer to User - Portal Preferences for more information about how users configure their zones.

Removing Zones From Portal

You cannot remove a base product zone from a base product portal. An implementation may override the **Display** setting to prevent a zone from displaying on the portal. In addition, you cannot remove a zone if a user has enabled it on their portal preferences. To remove a zone from the portal list, first make sure that no user has it enabled in their portal preferences.

Granting Access to A Portal

An application service is associated with each portal. A user must be granted access rights to the respective application service in order to see a portal.

Copyright © 2024, Oracle and/or its affiliates.

When you add a new **Standalone** portal, the system automatically creates its application service behind the scenes and displays it as part of the portal definition. You'll need to use this application service to grant access to the portal. **Tab Page** portals do not have separate security access. If a user has access to the main stand-alone portal, then the user will have security access to all its tabs.

Please note the following in respect of how application security impacts a user's portals:

- A user's Portal Preferences page only shows the portals configured to show on user preferences and where they have security access.
- The system's menus only show portals to which a user has security access.
- Users can set up favorite links to all portals, but they must have security rights to the portal's application service in order to invoke the favorite link.

Granting Access to Zones

An application service is associated with each zone. A user must be granted access rights to the respective application service in order to see a zone on a portal. Refer to The Big Picture Of Application Security for information about granting users access rights to an application service.

Please note the following with respect to zone application security:

- For most base product portals, all the zones for all the portals reference the same application service that is used to grant access to the portal. In other words, if the user has access to the page, then the user has access to all the zones on all portals for the page. There may be exceptions to this rule for certain portals.
- For a base product multi-query zones, typically the individual query zones and the multi-query zone reference the same application service that is used to grant access to the main (standalone) portal for the page. However, there may be individual query zones provided with a unique application service. This may occur when the query option is unusual and not applicable to all users or even to all implementations. If a user does not have security access to an individual query zone, that option will not be available in the dropdown.
- For base product portals that are configured to show on portal preferences, it is common that the portal contains different types of zones that may be applicable to different types of users. Typically these types of portals will deliver a unique application service for each zone so that an implementation may configure which user groups are allowed to view each zone. For these types of portals, please note the following:
    - A user's Portal Preferences page contains a row for a zone regardless of whether the user has access rights to the zone. Because of this, the system displays an indication of the user's access rights to each zone.
    - If a user's access rights to a zone are revoked, the zone will be suppressed when the user navigates to the respective portal.
    - Revoking a user's access rights does not change the user's portal preferences (i.e., a user can indicate they want to see a zone even if they don't have access to the zone - such a zone just won't appear when the respective portal appears).

Putting Portals on Menus

When you add a new **Standalone** portal, the system automatically creates its navigation option behind the scenes and displays it as part of the portal definition. You'll need to use this navigation option to add the portal to a menu or to allow any other type of navigation method to the portal.

To add a portal to a menu, you must add a menu item to the desired menu and reference the portal’s navigation option on the new menu item. A portal's navigation option can appear on any number of menu items (i.e., you can create several menu items that reference the same portal).

Your users can set up their preferences to include the portal's navigation option on their Favorite Links. This way, they can easily navigate to the portal without going through menus.

Portal Options

Various portal features are controlled by configuration options, such as the ability to include additional applicable actions on the portal’s actions bar or the ability to include the standard information description of the maintained entity as part of the portal’s title.

Copyright © 2024, Oracle and/or its affiliates.

> **Note: You can add new option types.** Your implementation may want to add additional portal option types. To do that, add your new values to the customizable lookup field **PORTAL_OPT_FLG**.

Setting Up Portals and Zones

The topics in this section describe how to set up portals and zones. Please refer to The Big Picture of Portals and Zones for background information.

Defining Zone Types

A Zone Types represents a particular type of zone with a specific behavior. For example, a data explorer zone type is used to select data using a specific SQL statement and display the data based on parameter configuration. The zone type defines the Java Class that controls the behavior of the zone and defines the parameters that the Java Class supports. The base product supports many zone types used to build the portal / zone user interface. Implementations may introduce their own zone types.

> **Note:** It is not very common for an implementation to introduce their own zone types.

Select **Admin** > **System** > **Zone Type** to maintain zone types.

Description of Page

Specify an easily recognizable **Zone Type** code and **Description**. Use the **Detailed Description** to describe in detail what the zone type does.

> **CAUTION:** When adding new zone types, carefully consider its naming convention. Refer to System Data Naming Convention for more information.

Owner indicates if this zone type is owned by the base product or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a zone type. This information is display-only.

Java Class Name is the Java class responsible for building the zone using the parameters defined below.

Two types of parameters are specified when defining a zone type:

- Parameter values that have a **Usage** of **Zone** are defined on the zones and control the functionality of each zone governed by the zone type. A **Usage** value of **Zone - Override Allowed** indicates that an implementation may override the parameter value for a base zone.
- Parameter values that have a **Usage** of **Zone Type** are defined directly on the zone type and control how the zone type operates (e.g., the name of the XSL template, the name of the application service). A **Usage** value of **Zone Type - Override Allowed** indicates that an implementation may override the parameter value for a base zone type.

The following points describe the fields that are defined for each parameter:

- **Sequence** defines the relative position of the parameter.
- **Parameter Name** is the name of the parameter.
- **Description** is a short description that allows you to easily identify the purpose of the parameter.
- **Comments** contain information that you must know about the parameter or its implementation. For parameters with a usage of **Zone** or **Zone - Override Allowed**, this information is visible to the user when viewing or defining this parameter for a zone of this type.
- **Usage** indicates whether the parameter value is defined in a **Zone** of this type or in the **Zone Type**. **Zone - Override Allowed** and **Zone Type - Override Allowed** indicate that override values for the parameters defined in a base zone or base zone type can be entered.
- **Required** is checked to indicate that a zone must define a value for the parameter. It is not checked if a value for the parameter is optional. This field is protected if the **Usage** is **Zone Type** or **Zone Type - Override Allowed**.

Copyright © 2024, Oracle and/or its affiliates.

- **Parameter Value** is used to define the value of zone type parameters. This field is protected if the **Usage** is **Zone** or **Zone - Override Allowed**.
- **Owner** indicates if this parameter is owned by the base product or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a parameter. This information is display-only.

Zone Type Parameter Comments

For the product owned zone type parameters, the parameter's detailed description provides the detail needed for properly configuring the parameter. For the Action parameters (IMPLEMENTOR_ACTION_n), the parameter description is abbreviated. Additional detail about configuring this parameter may be found in the Zone Action Parameter detailed information. The same details apply.

Defining Zones

This portal is used to view and maintain the configuration of a zone. Implementations may define custom zones or override the descriptions and some parameter values for base product zones.

You can access the portal from the**Admin Menu** > **System** > **Zone**. You are brought to a query portal with options for searching for a specific zone. Once a zone record has been selected you are brought to the maintenance portal to view and maintain the selected record.

Note: When introducing a new record, carefully consider its naming convention. Refer to System Data Naming Convention for more information.

Refer to Understanding Portals and Zones for more information.

The following zones may appear as part of the portal's **Main** tab page

- **Zone.** This zone provides general information about the selected zone.
- **Query Options.** This zone maintains the list of query options for a Multi-Query zone. The zone is only visible for Multi-Query zones.
- **Filters.** This zone maintains various zone parameters that handle filtering options for a data explorer zone. The zone is only visible for data explorer zones.
- **General Parameters.** This zone maintains various general purpose zone parameters for a data explorer zone.
- **SQLs.** This zone maintains individual SQLs and their related parameters. The zone is only visible for data explorer zones.
- **SQL Configuration.** This zone maintains all the zone parameters related to a specific data explorer SQL. The zone is only visible when a specific SQL is broadcast from the **SQLs** zone.

The following zones may appear as part of the portal's **References** tab page

- **Zone References.** This zone list various configuration entities referencing the current zone.

Overriding Parameter Values

Implementation teams may override the value of selected types of base provided zone parameters. The override parameter value field for base owned zone parameters is only enabled when the related zone type parameter usage is **Zone - Override Allowed**.

For multi-query zones, you may use the override parameter value to add custom query options to a base product owned zone.

Overriding Zone Parameters

Implementations may override the value of a base owned zone parameter only if the zone type definition allows such override. The override parameter value is enabled for a specific base owned parameter when the related zone type parameter usage is set to **Zone - Override Allowed**.

Copyright © 2024, Oracle and/or its affiliates.

You may use the override parameter capability to provide alternative or additional custom query options for a base owned multi-query zone.

Zone Parameter Details

For most zone parameters, the embedded help for the parameter provides the detailed information needed for configuring the parameter values. For some parameters with very detailed descriptions, the embedded help is abbreviated and more detail is provided here.

Zone Visibility Service Script

All zones support a visibility script that is used to determine if the zone should be displayed to the user or not based on conditions. The script may receive input parameters and is expected to return a Boolean value indicating if the zone should be displayed or not. The embedded help for the **Zone Visibility Service Script** parameter provides details related to the syntax.

The following table highlights some service scripts provided by the product that may be used if applicable to your zone's requirements. This is not an exhaustive list of visibility scripts. There may be others that are specific to a given zone.

Script Code	Description	Comments
F1-ShldShwZn	Zone Visibility - Display Zone in Portal	This script simply returns a value of 'true' and is used when the zone should always appear.
F1-CondShwZn	Zone Visibility - Display Zone in Portal Conditionally	This is used when the condition for showing the zone is based on the population of a context value. This is commonly used when one zone in the portal should only appear after a broadcast of a record from another zone in the portal. For an example of a zone the uses this visibility script, refer to **F1-BSFTYPE** .
F1-RwCtShwZn	Zone Visibility - Based on Row Count	This is used when the condition for showing the zone is based on the existence of one or more rows that can be determined using SQL. This script accepts a zone code, user filters 1 through 25 and hidden filters 1 through 10. The script returns an indication of 'true' if at least one row count is returned by the zone. To use this visibility script, a specific data explorer zone must be developed for the specific use case. For an example of a zone the uses this visibility script, refer to **F1-MIGRREQEL** .

SQL Statement

Data explorer zones are used to select data to display using one or more SQL statements. The SQL parameters are applicable to the following zone types

- Info Data Explorer - Single SQL (**F1–DE-SINGLE**). The parameter has the description **SQL Statement**.
- Info Data Explorer - Multiple SQLs (**F1–DE**). The parameters follow the description pattern of **SQL Statement x**.
- Query Data Explorer - Multiple SQLs (**F1–DE-QUERY**). The parameters follow the description pattern of **SQL Statement x**.

Note: If your implementation has been configured to restrict the functions that may be used when defining an SQL then an error is issued at runtime if there are functions found that are not in the allowlist. The allowlist may be viewed using the **View SQL function allowlist** link in the Tips zone on the zone maintenance page.

Copyright © 2024, Oracle and/or its affiliates.

Note: Zone SQL should take care not to use the following reserved keywords for the table alias. (For example, the table alias in this SQL is 'XX': SELECT 'X' FROM W1_ACTIVITY_TYPE **XX**.)

Keyword
AT

The following table provides a list of SQL substituted keywords that may be used in the SQL Statement parameters in explorer zones. At execution time, the system determines the database and substitutes the keyword with the database specific syntax:

Keyword	Description	Examples
@toCharacter()	Converts the input to Character data type.	select **@toCharacter**(batch_cd) as batchCode from ci_batch_ctrl
@toDate()	Converts the input to Date data type.	select **@toDate**(last_update_dttm) as lastUpdateDate from ci_batch_ctrl
@toNumber()	Converts the input to Number data type.	select **@toNumber**(next_batch_nbr) from ci_batch_ctrl
@currentDate	Fetches the current date. **CAUTION:** The Oracle functions **SYSDATE** and **CURRENT_DATE** should not be used because they do not properly cater for adjusting dates from the database time zone to the installation time zone, if needed.	select batch_cd, **@currentDate** as today from ci_batch_ctrl
@currentTimestamp	Fetches the current date / time. **CAUTION:** The Oracle functions **SYSTIMESTAMP** and **CURRENT_TIMESTAMP** should not be used because they do not properly cater for adjusting the date / time from the database time zone to the installation time zone, if needed.	select batch_cd from ci_batch_ctrl where last_update_dttm > **@currentTimestamp**
@concat	Combines the result list of two or more columns.	select batch_cd **@concat** next_batch_nbr concatNbr from ci_batch_ctrl
@substr(string, start)	**String** is the input String that you are trying to get a substring of. **Start** is the position of the character for the output results.	select batch_cd batchCode from ci_batch_ctrl Result: TESTCD select **@substr**(batch_cd,3) batchCode from ci_batch_ctrl Result: STCD
@substr(string, start, end)	**String** is the input String that you are trying to get a substring of.	Select batch_cd batchCode from ci_batch_ctrl

Copyright © 2024, Oracle and/or its affiliates.

Keyword	Description	Examples
	Start is the position of the character for the output results. **End** is the number of characters required in the output from starting position.	Result: TESTCD select **@substr**(batch_cd,3,2) batchCode from ci_batch_ctrl Result: ST
@trim	Trims the white spaces of the output on both sides.	select **@trim**(batch_cd) as batchCode from ci_batch_ctrl
The following syntax is related to 'fuzzy' searching. It is only applicable if Oracle DB Text is enabled and a context text index has been created. Refer to Advanced Search Options for more information.		
@fuzzy(string, score, numresult, 'weight')	**String** is the input value for the search. **Score** is the degree of 'fuzziness'. Valid values are between 1 - 80. The higher the number the more precise the search. Default is 60. **Numresults** is the number of variations to consider for the string. Valid values are between 1 and 5000. Default is 100. Indicate **'weight'** to signal that the results are returned in order of weight. Leave this setting off to indicate that the results are returned in order of score.	Set score to 70, number results to 6, and specify weight. select user_id, last_name from sc_user where contains(last_name, @fuzzy(:F1,70, 6, 'weight')) > 0
@fuzzy(string)	This returns a string result from the fuzzy expansion operation where the default value of 60 is assumed for the **score** and the default value of 100 is assumed for the **numresult**.	To use default values: select user_id, last_name from sc_user where contains(last_name, @fuzzy(:F1))> 0
@fuzzy(string, score)	This returns a string result from the fuzzy expansion operation with the **score** specified and the default value of 100 for the **numresult**.	Set score to 70. select user_id, last_name from sc_user where contains(last_name, @fuzzy(:F1,70)) > 0
@fuzzy(string, score, numresult)	This returns a string resulted from the fuzzy expansion operation with the similarity score and the **numresults** specified.	Set score to 70, number results to 6. select user_id, last_name from sc_user where contains(last_name, @fuzzy(:F1,70, 6)) > 0

Column Parameters

eData explorer zones are used to select data to display using one or more SQL statements. For each SQL statement, the zone may configure up to 20 Columns that contain the formatting definition for displaying the output data.

These parameters are applicable to the zone types

- Info Data Explorer - Single SQL (**F1-DE-SINGLE**). The parameters follow the description pattern of **Column x**.
- Info Data Explorer - Multiple SQLs (**F1-DE**). For this zone type, all SQLs are executed and the zone displays a union of all the results. The parameters follow the description pattern of **Column x for SQL y**. There are some mnemonics that do not make sense to differ within rows of the same column. For example, the column label. For

Copyright © 2024, Oracle and/or its affiliates.

these mnemonics, only the value in SQL 1 is considered for that column. The table below indicates which mnemonics follow this rule.

- Query Data Explorer - Multiple SQLs (**F1-DE-QUERY**). For this zone type, only one SQL is executed. The **SQL y Condition** parameter may be used to control this. The system will execute the first SQL whose condition is satisfied (or with no condition populated). The parameters follow the description pattern of **Column x for SQL y**.

The following sections describe the various types of mnemonics.

Source Mnemonics

This table describe the mnemonics that control how the data in a column is derived.

Mnemonic	Description	Valid Values	Comments
source=	Defines how the column's value is derived.	**SQLCOL**	Indicates that the source of the column's value comes from a column in the SQL statement. This type of column must also reference the **sqlcol=** mnemonic.
		BO	Indicates that the source of the column's value comes from a business object. This type of column must also reference the **bo=**, **input=** and **output=** mnemonics to define how to interact with the business object.
		BS	Indicates that the source of the column's value comes from a business service. This type of column must also reference the **bs=**, **input=** and **output=** mnemonics to define how to interact with the business service.
		SS	Indicates that the source of the column's value comes from a service script. This type of column must also reference the **ss=**, **input=** and **output=** mnemonics to define how to interact with the service script.
		FORMULA	Indicates that the source of this column's value is calculated using a formula. This type of column must also reference the **formula=** mnemonic.
		SETFUNC	Indicates that the source of this column's value is calculated using a superset of values from the rows in the SQL statement. This type of column must also reference the **setfunc=** mnemonic.
		ICON	Indicates that the source of this column's value is a display icon reference (meaning that an icon will be displayed in the column). This type of column must

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Description	Valid Values	Comments
			also reference the **icon=** mnemonic to define the icon reference.
			Note: When using this source mnemonic, the formatting mnemonic **type=** is not applicable.
		FKREF	Indicates that the source of this column's value is an FK reference (meaning that the FK reference's context menu and information string will be displayed in the column and will be enabled for navigation). This type of column must also reference the **fkref=** and **input=** mnemonics to define how the FK reference is called.
			Note: When using this source mnemonic, the formatting mnemonic **type=** is not applicable.
			Note: To show a foreign key's information but not enable navigation, refer to Common Use Cases for the recommended technique.
		SPECIFIED	Indicates that the source of this column's value is specified by concatenating literals and other column values. This type of column must also reference the **spec=** mnemonic.
		MSG	Indicates that the source of this column is a message from the message table (along with any substitution variables). This type of column must also reference the **msg=** mnemonic.
sqlcol=	Defines the column in the SQL statement when **source=SQLCOL**.	COLUMN_NAME	Enter the name of a column that is retrieved in the SELECT statement. Note that if the select statement uses an alias for a column, then the alias should be referenced here.
		x	Where x is an integer value that references a column by its relative position in the SELECT statement. For example, **sqlcol=3** would display the 3rd column in the SELECT statement).
bo=	Defines the business object to invoke when **source=BO**.	'Business Object Code'	

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Description	Valid Values	Comments
	This mnemonic must be used in conjunction with the **input=** and **output=** mnemonics to define how information is sent to / received from the business object.		
bs=	Defines the business service to invoke when **source=BS**. This mnemonic must be used in conjunction with the **input=** and **output=** mnemonics to define how information is sent to/ received from the business service.	'Business Service Code'	
ss=	Defines the service script to invoke when **source=SS**. This mnemonic must be used in conjunction with the **input=** and **output=** mnemonics to define how information is sent to / received from the service script.	'Service Script Code'	
fkref=	Defines the FK reference used to retrieve the column's information when **source=FKREF**. This mnemonic must be used in conjunction with the **input=** mnemonic to define how information is sent to the FK reference to build the information.	Cx	This means FK reference code is defined in an earlier column. For example, define **C1** if column 1 defines the FK reference value.
		COLUMN_NAME	This means the FK reference was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause.
		'FK Reference Code'	This means the FK Reference is defined directly. For example 'F1-ROLE'.
formula=	Defines the formula to use when **source=FORMULA**. Examples: • **formula=C1*.90/C2** • **formula=(C1/C2)*100**	The formula can contain numeric constants, operators and column references.	For column references, use the format Cx where x represents the column number.
			Refer to Expression Parser for information about the functions supported.
setfunc=	Defines the function to apply to the rows of	function(Cx)	Where Cx represents a column whose rows should have the function applied and the function is one of the following:

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Description	Valid Values	Comments
	a given column when **source=SETFUNC**.		• **MAX**. This derives the maximum value of all rows in the column. • **MIN**. This derives the minimum value of all rows in the column. • **TOT**. This derives the sum (total value) of all rows in the column. • **ACC**. This derives the cumulative total of all rows up to an including the current row.
input=	This is used to define one or more input fields and values passed to business objects, business services, service scripts, and FK references. The syntax is as follows: [ELEMENT_NAME=ELEMENT_REF ELEMENT_NAME=ELEMENT_REF ...] In other words, the list of input values is surrounded by square brackets separated by a space. Each passed value first defines the ELEMENT_NAME, which is the name of the element / field in the target. ELEMENT_REF is the value passed in. The next column indicates the possible values for ELEMENT_REF.	Cx	Where Cx represents the value of a previous column. If the value to pass is in the first column, reference C1.
		COLUMN_NAME	This means the value to pass in was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause.
		'literal value'	This means a literal value within the single quotes should be passed in.
		userTimeZone	This means the current user's time zone should be passed in. This is typically used with the business service **F1-ShiftDateTime** to convert data in the storage time zone to the user's time zone for display.
		installationTimeZone	This means the installation time zone should be passed in. This is typically used with the business service **F1-ShiftDateTime** to convert data in the storage time zone to the installation time zone for display.
		Examples: • **input=[USER_ID=C1]** • **input=[USER_ID=USER_ID]** • **input=[input/targetTimeZone=userTimeZone]**	
output=	This is used to define the name of the element retrieved from the business object, business service or service script used to populate this column.	elementName	Example: **output=personInfo**
pagingkey=	This mnemonic is only applicable when the **Enable Paging** parameter has been configured. It indicates that this column is one of the keys used by the SQL statement to	**Y**	
		N	This is the default, meaning that you don't need to indicate **pagingkey=N** at all to indicate that the column is not one of the paging keys.

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Description	Valid Values	Comments
	orchestrate paging through results. This mnemonic can only be specified when the **source=SQLCOL**.		
	Fastpath: Refer to Pagination Configuration for more information.		

Note: If multiple columns are configured with the same source BO, BS or SS and the same input data, the system caches the output from the first call and reuses the results for subsequent columns.

Formatting Mnemonics

This table describe the mnemonics that control how a column is formatted.

Note: For the **F1-DE** zone type, zone displays the union of all the different SQLs.

Mnemonic	Description	Valid Values	Comments
type=	Defines how the column's value is formatted.	**STRING**	Columns of this type capture a string. This is the default value.
	Note: Icon and Foreign Key columns. The **source=** source mnemonic may be used to indicate a column should be derived from an icon reference or a foreign key (FK) reference. If you use either of these sources, the **type=** mnemonic is not relevant as either an icon or a context menu / info string will appear in the column.	**DATE**	Columns of this type capture a date and will be displayed using the user's display profile.
		TIME	Columns of this type capture a time (in database format) and will be displayed using the user's display profile.
		DATE/TIME	Columns of this type capture a date and time (in database format) and will be displayed using the user's display profile.
		MONEY	Columns of this type capture a monetary field. This type of column may also reference the **cur=** mnemonic. If the cur mnemonic is not specified, the currency code on the installation record is used.
		NUMBER	Columns of this type capture a numeric field. This type of column may also reference the **dec=** mnemonic.
label=	Defines the column's override label. The label appears in the column's heading and in the zone's drag and drop area.	FIELD_NAME	Enter a valid field name whose label should be used for the column label. This should always be the option used if multiple languages are needed.
		'text'	Defines the text directly.

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Description	Valid Values	Comments
	If this mnemonic is not defined, the system uses the column's default label. The source of a column's default label differs depending on the column's source. Note that some sources don't have a default value and omitting this mnemonic will result in a blank label. **Note:** In the case of the zone type Info Data Explorer - Multiple SQLs (**F1–DE**), the system will use the column definitions in SQL 1 to define the labels.		
cur=	Defines the currency code applied when **type=MONEY** if the installation record's currency should not be used.	Cx	This means currency code value is defined in an earlier column. For example, define **C1** if column 1 defines the currency code.
		COLUMN_NAME	This means the currency code was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause.
		'Currency Code'	This means the currency code is defined directly. For example 'USD'.
dec=	Defines the number of decimal places when **type=NUMBER**. It is optional. If provided it should be an integer. If not provided, the number of decimals will default to the number of decimal places defined on the currency code	nR	Where n is the number of decimal places to show. Suffixing the number of decimal places with R means that the system should round up / down. Simply specifying n (without an R) means that decimal places should be truncated. For example, entering **dec=4** will display 4 decimal places and truncate the remainder.

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Description	Valid Values	Comments
	specified on the installation record.		**Note: Formatting only.** This mnemonic is only used for formatting, it does not impact the precision used for subsequent calculations. For example, if a column retrieved from the database contains 6 significant digits and **dec=0**, the column will be shown with no decimal places (truncated), however any references to the column in subsequent calculations will use 6 decimal places. For example, if the column is referenced in a formula or set function, all 6 decimal places will be used.
char=	This mnemonic applies special character(s) to the column's value.	'x[]x'	Where x references the literal value to display and **[ ]** defines the relative position of the characters (before or after the value). You need only include the **[ ]** if you want to position characters in front of the value. For example, **char='%'** will place a percent sign after the value. If you want to position the word 'minutes' before a value, enter **char='minutes [ ]'**. If you want to output a value like **BUDGET $123.12 (YTD)**, enter **char='BUDGET [ ] (YTD)'**.
suppress=	This is used to indicate a column should not be displayed. A column would be suppressed if it's only defined for use by subsequent columns, for example, if there is a formula that derives a column using two other columns. In this scenario, the columns referenced in the formula can be suppressed.	**true**	
	Note: In the case of the zone type Info Data Explorer - Multiple SQLs (**F1–DE**), the system will apply the settings for the column definitions in SQL 1 to all subsequent SQLs.	**false**	This is the default, meaning that you don't need to indicate **suppress=false** at all to indicate that the field should be shown.

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Description	Valid Values	Comments
suppressSearch=	This is used to indicate a column should not be displayed when the zone is invoked in search mode only. **Note:** In the case of the zone type Info Data Explorer - Multiple SQLs (**F1–DE**), the system will apply the settings for the column definitions in SQL 1 to all subsequent SQLs.	**true**	
		false	This is the default, meaning that you don't need to indicate **suppressSearch=false** at all to indicate that the field should be shown.
suppressExport=	This is used to indicate a column should not be downloaded when exporting. **Note:** In the case of the zone type Info Data Explorer - Multiple SQLs (**F1–DE**), the system will apply the settings for the column definitions in SQL 1 to all subsequent SQLs.	**true**	
		false	This is the default, meaning that you don't need to indicate **suppressExport=false** at all to indicate that the field should be included in a download.
width=	This is used to override the width of a column (number of pixels). The default value is the maximum width of any cell in the column.	n	Where n is a number between 0 and 999. **Note:** If there is no available breaking point in the data, the column will be longer than the specified number of pixels. The length of the column's label (which appears in the column's heading) may also make the width wider than specified. In the case of the zone type Info Data Explorer - Multiple SQLs (**F1–DE**), the system will set the width based on the column definitions in SQL 1.
color=	This is used to override the column's text color.	A valid HTML "named" color	For example **color=yellow**.

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Description	Valid Values	Comments
			Note: Refer to Color Contrast for information about the use of the HTML color 'red' and its impact on accessibility.
		A valid RGB color model combination	For example **color=#E0292F** or **color=#CCCCCC**. Note that the # is required.
bgcolor=	This is used to override the column's background color.	A valid HTML "named" color	Similar to the **color=** mnemonic.
		A valid RGB color model combination	Similar to the **color=** mnemonic.
order=	Defines the column's default sort order.	**ASC**	Indicates that the order is ascending. This is the default meaning that it is not necessary to indicate **order=ASC**.
		DESC	Indicates that the order is descending.
rowHeader=	Designates the column as a row header for accessibility purposes.	**true**	By default, the first data column of the data explorer results is identified as the row header, for accessibility tools. If the data in the first column does not uniquely identify the row, use this mnemonic to explicitly mark a different column or multiple columns as the row header.
			Note: In the case of the zone type Info Data Explorer - Multiple SQLs (**F1–DE**), the system will look only at the columns in SQL 1 that have this mnemonic defined and apply it to all results for those columns.

Click Mnemonics

This table describe the mnemonics that define whether a column value may be clicked and if so, what should happen.

Mnemonic	Description	Valid Values	Comments
navopt=	Defines the navigation option that references the target transaction or script when the user clicks a column. Note, this mnemonic should be used in conjunction with the **context=** mnemonic to define what information	Cx	This means navigation option code is defined in an earlier column. For example, define **C1** if column 1 defines the navigation option.
		COLUMN_NAME	This means the navigation option was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause. Example: **navopt=MAIN_PORTAL**

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Description	Valid Values	Comments
	is sent to the navigation option's target transaction. This mnemonic is ignored if **source=FKREF** because the FK reference code defines the hyperlink's destination.	'Navigation Option Code'	This means the navigation option code is defined directly. For example **navopt='userMaint'**.
context=	This is used to define one or more context fields and values passed to the target navigation option to go along with the **navopt=** mnemonic. The syntax is as follows: [FIELD_NAME=FIELD_REF FIELD_NAME=FIELD_REF ...] In other words, the list of input values is surrounded by square brackets separated by a space. Each passed value first defines the FIELD_NAME, which is the name of the context field in the navigation option. FIELD_REF is the value passed in. The next column indicates the possible values for FIELD_REF.	Cx	Where Cx represents the value of a previous column. For example, if the value to pass is in the first column, reference **C1**.
		COLUMN_NAME	This means the value to pass in was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause.
		'literal value'	This means a literal value within the single quotes should be passed in.
bpa=	Indicates that a BPA script should be executed with the user clicks the column and indicates the BPA to execute. Note, this mnemonic should be used in conjunction with the **tempstorage=** mnemonic to define the temporary storage values that will be initiated when the script is executed. This mnemonic is ignored if **source=FKREF** because the FK reference code defines the hyperlink's destination.	Cx	Indicates that the BPA script is defined in a previous column.
		COLUMN_NAME	This means the BPA script to execute was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause.
		'BPA Script Code'	This means that the BPA script to execute is defined directly.

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Description	Valid Values	Comments
tempstorage=	This is used to define how temporary storage variables are initiated when the **bpa=** mnemonic is used. The syntax is as follows: [FIELD_NAME=FIELD_REF FIELD_NAME=FIELD_REF ...] In other words, the list of input values is surrounded by square brackets separated by a space. Each passed value first defines the FIELD_NAME, which is the name of the field in temporary storage. FIELD_REF is the value passed in. The next column indicates the possible values for FIELD_REF.	Cx	Where Cx represents the value of a previous column. For example, if the value to pass is in the first column, reference **C1**.
		COLUMN_NAME	This means the value to pass in was retrieved by the SELECT statement. The value should match the name defined in the SELECT clause.
		'literal value'	This means a literal value within the single quotes should be passed in.
list=	This is used to enable work list capability for this column. You may optionally populate the **listdesc=** mnemonic to override the text that will be placed in the worklist zone.	**true**	Setting **list=true** will cause the work list icon to appear in the column's header. If a user clicks the column, it will populate all the rows in the output into the work list zone. **Note:** In the case of the zone type Info Data Explorer - Multiple SQLs (**F1-DE**), the output may be showing a union of the results of multiple SQL statements. In this case, if some of the SQL statements configure a given column with **list=true**, but not all, only the data in the cells for the statements that configure this mnemonic are put into the work list when the user clicks the icon. Also note that when determining which columns should have the worklist icon when building the zone, the system only looks at the configuration for the columns in SQL 1.
listdesc=	This is an optional mnemonic when using the **list=** mnemonic. It can be used to override the text that is placed in the work list zone.	Cx	Where Cx represents the value of a previous column. For example, if the text to use is in the first column, reference **C1**.

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Description	Valid Values	Comments
listbroadcast=	Indicates that the broadcast information for the column is also to be made available in the work list zone. This means that the work list can be used to broadcast information to a portal in the same manner as a data explorer.	**true**	Use this setting to turn on the feature.

Common Use Cases

This section provides some common configuration that can be copied and pasted for newly created zones.

Use Case	Code Snippet
Displaying the description of a lookup value that has been retrieved in the SQL statement. The business service checks for an override label.	**source=BS** **bs='F1-GetLookupDescription'** **label=FLD_NAME** **input=[fieldName='FLD_NAME' fieldValue=FLD_NAME]** **output=description**
Display a foreign key with its information but suppressing the navigation. (This is useful when you have a foreign key that is additional information in a zone but would cause confusion to enable navigation. To display foreign key reference with its information and enabled for navigation, use the source of FKREF as described above.) The first example assumes that you know the FK Reference. The second example uses a business service that receives the maintenance object as input.	**source=BS** **bs='F1-GetForeignKeyReference'** **input=[input/fkReference=FK_REF_CD input/fkValue1=BUS_OBJ_CD]** **output=output/infoDescription** **label=DESCR**
	source=BS **bs='F1-GetFKReferenceDetails'** **input=[input/maintenanceObject='F1-EXT_LKUP' input/pkValue1=BUS_OBJ_CD input/pkValue2=C1]** **output=output/infoDescription** **label=DESCR**
Display a foreign key, including a corresponding navigation link, for generic objects that may or may not have a foreign key reference. If the FK reference exists, its information is shown, otherwise not. Standard foreign key services assume all rows have foreign key details. The service script **F1FKInfoOpt** provides the same foreign key details but does not error when foreign key details do not exist. To build the foreign key description and link, the script needs	This column gets the navigation details: **source=SS** **ss='F1FKInfoOpt'** **input=[input/fkReference=C9 input/fkValue1=C6 input/fkValue2=C7]** **output=output/navigationOption** **suppress=true**

Copyright © 2024, Oracle and/or its affiliates.

Use Case	Code Snippet
to be used twice; one column gets the navigation details and another gets the description. The latter references the navigation details column and invokes a BPA script to perform the dynamic navigation.	This column constructs the foreign key description and link: **source=SS** **ss='F1FKInfoOpt'** **input=[input/fkReference=C9 input/fkValue1=C6 input/fkValue2=C7]** **output=output/infoDescription** **label=F1_APPSEC_COMP_DESCR** **bpa='F1ApsecNav'** **tempstorage=[navopt=C15 pkField1=C10 pkValue1=C6 pkField2=C11 pkValue2=C7]** **list=true**

Zone Action

Most zone types provided by the product allow for one or more Zone Actions to be defined to appear in the zone header. An action can appear as a hyperlink, icon or button. The action can also be provided as an HTML string.

Note: Zone types also include parameters for actions defined at the zone type level using IMPLEMENTOR_ACTION_n (Action n) parameters. These are rarely used by the product zone types. The actions defined here override any actions defined on the zone type (if present). The details below apply to the zone type level actions as well.

A zone action is defined using the following mnemonics:

Mnemonic	Description	Valid Values	Comments
type=	This mnemonic defines the appearance of the action in the zone header.	**LINK**	Indicates that the action is shown as a textual hyperlink.
		ICON	Indicates that the action is shown as a graphical icon.
		BUTTON	Indicates that the action is shown as an HTML button.
		ASIS	Indicates that the parameter will provide

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Description	Valid Values	Comments
			the HTML to be used for the action.
action=	This mnemonic defines the action to take when the link/icon/button is clicked. This is ignored when the type=ASIS.	**NAVIGATION**	Indicates that the action is navigation to a page.
		SCRIPT	Indicates that the action is to run a BPA script.
navopt=	Defines the navigation option to use when the **action=NAVIGATION**.	'NAV_OPT_CD'	Enter a reference to a valid navigation option in single quotes.
bpa=	Defines the script to run when the **action=SCRIPT**.	'SCRIPT_CD'	Enter a reference to a valid BPA script in single quotes.
icon=	Indicates the icon to use when **type=ICON**.	DISP_ICON_CD	Enter a reference to a valid display icon.
		'path'	Enter an explicit path to the icon, for example 'images/gotoZone.gif'
asis=	This is required when the **type=ASIS**. This provides the ability to precisely define the HTML you wish to have included in the header. All valid HTML is permitted including the use of "ora" css classes and JavaScript functions.	['HTML']	

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Description	Valid Values	Comments
label=	By default, the label or tooltip will come from the navigation option or BPA script description. Use this mnemonic to override that label.	FIELD_NAME	Enter a valid field name whose label should be used. This should always be the option used if multiple languages are needed.
		'text'	Enter the text directly in single quotes.
context=[target1=source1 target2=source2]	This is used to pass context data when navigating to a page or executing a BPA script. The mnemonic supports passing multiple values. In each case the target context field or BPA script variable is defined first followed by an equal sign, followed by source data defined using one of the valid values defined in the next column. One or more values may be defined. Each context value is defined separated by spaces. The whole set of context values should be surrounded by square brackets.	FIELD_NAME	Indicates that the value should be taken from the field with this name from portal context, global context or the page data model. The mnemonic sourceLoc is used for defining the source.
		xpath	Indicates that the value should be taken from a schema field, represented by the Xpath, displayed in this zone. This is valid when the zone is

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Description	Valid Values	Comments
			displaying a UI Map.
		'constant'	Indicates that the value defined in single quotes should be passed.
sourceLoc=	This mnemonic defines the source of the FIELD_NAME's value in the context mnemonic. If this mnemonic is left blank, the default behavior is as follows: - The portal context is checked. - If no portal context value is found, the global context is checked. - If neither value is available, the field is ignored.	**G**	Indicates that the field's value is retrieved from the global context.
		P	Indicates that the field's value is retrieved from the portal context.
		D	Indicates that the field's value is retrieved from the page data model.
class=	Use this mnemonic to override the look and feel of the link / icon / button using a different CSS style.	'className1' 'className2'	Enter one or more classes in single quotes. Multiple class names may be provided.
style=	Use this mnemonic to override the look and feel of the action element using the indicated css style.	Standard style= format.	All allowed css style definitions may be used.

Examples:

- **type=BUTTON action=SCRIPT bpa='F1-SET-USER' context=[USER_ID=USER_ID] label=UPDATE_LBL**

Copyright © 2024, Oracle and/or its affiliates.

- **type=LINK action=NAVIGATION navopt='gotoUser' context=[USER_ID=path(schema/userdId)]**
- **type=ASIS asis=['<A class="oraLink" href="www.google.com">Search']**

> **Note:** If the zone type has actions defined and there is a desire to simply remove the zone type actions, the Zone Action can be set with the following configuration: **type=ASIS asis=[]**

User Filters

Data explorer zones include the ability to define User filters to allow a user to enter data to restrict the zone's rows and / or columns. The filters may be defined individually using User Filter parameters 1–25. Alternatively, a UI map may be defined for capturing filters. In this case, the map's input fields must be associated with the zone's filters by specifying the **xpath=** mnemonic on the respective User Filter parameters.

These parameters are applicable to the zone types

- Info Data Explorer - Multiple SQLs (**F1–DE**)
- Query Data Explorer - Multiple SQLs (**F1–DE-QUERY**)
- Info Data Explorer - Single SQL (**F1–DE-SINGLE**)

A user filter is defined using the following mnemonics:

Mnemonic	Description	Valid Values	Comments
name=	This mnemonic is used if the zone's filter should be pre-populated with a value from global context, portal context or broadcast from another zone.	MD Field Name	
datasource=	This mnemonic defines the source of the filter's pre-populated value defined in the name mnemonic. If this mnemonic is left blank, the default behavior is as follows: - If the field has been broadcast from another zone, the broadcast value is used. - If no value is broadcast, the portal context is checked to determine if this field exists (if so, its value is taken). - If still no value, the global context is checked. - If still no value, no default value is shown.	**G**	Indicates that the zone should look for the filter value in global context.
		P	Indicates that the zone should look for the filter value in portal context.
		D	Indicates that the zone should look for the filter value in the page data model.
type=	Defines the visual metaphor used to capture the filter values.	**DATE**	Filters of this type capture a date.
		DATE/TIME	Filters of this type capture a date and time.

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Description	Valid Values	Comments
		STRING	Filters of this type capture a string
		MONEY	Filters of this type capture a monetary field. This type of filter must also reference the cur mnemonic.
		NUMBER	Filters of this type capture a numeric field. This type of filter may also reference the decimals mnemonic.
		LOOKUP	Filters of this type capture a lookup value. This type of filter must also reference the lookup mnemonic.
		TABLE	Filters of this type capture an administrative table's value (code and description). This type of filter must also reference the table mnemonic.
		CHARTYPE	Filters of this type capture predefined characteristic values for a characteristic type (code and description). This type of filter must also reference the chartype mnemonic.
		ASIS	Filters of this type capture a list of values to be referenced within an 'IN' clause within the SQL statement.
label=	Defines the filter's label that appears in the zone's description bar and in the input area.	MD Field Name	Enter a valid field name whose label should be used for the filter label. This should always be the option used if multiple languages are needed.
		'text'	Defines the text directly.
cur=	Defines the currency code applied when **type=MONEY**.	Currency Code	Enter a reference to a valid currency code.
dec=	Defines the number of decimal places when **type=NUMBER**.	Valid number	It is optional. If provided it should be an integer. If not provided, the number of decimals will default to the number of decimal places defined on the currency code specified on the installation record.
lookup=	Defines the lookup flag whose values appear when **type=LOOKUP**.	Lookup Field Name	Enter a reference to a valid lookup field name.

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Description	Valid Values	Comments
table=	Defines the admin table whose values appear when **type=TABLE**.	Table Name	Enter a reference to a valid control table name.
chartype=	Defines the characteristic type code whose values appear when **type=CHARTYPE**.	Char Type code	Enter a reference to a valid characteristic type code.
xpath=	This mnemonic is used in conjunction with a Filter Area UI Map. For each filter, you must specify the XPath to the corresponding UI map schema element.	XPath	The **type=** mnemonic must also be appropriate for the map's input field, otherwise the query's SQL could fail.
likeable=	This mnemonic defines if a likeable search is performed on the entered value when **type=STRING**.	**S**	The query will add % to the suffix of the filter value.
		P	The query will add % to the prefix of the filter value.
		PS	The query will add % to the prefix and suffix of the filter value.
divide=	The mnemonic controls if a divider line appears above and/or below the filter. Note, you can specify this parameter twice if you want divider lines placed above and below a filter, e.g., **divide=above divide=below**.	**above**	This results in a divider line placed above the filter.
		below	This results in a divider line placed below the filter.
searchField=	This mnemonic controls the initial population of the filter when the zone is launched as a search from a UI map.	MD Field Name	Enter the field name that exactly matches the searchField name specified in the oraSearchField HTML element in the UI map.
encrypt=	This mnemonic defines if the user filter is encrypted and needs to be searched by hashed value.	[TBL_NAME,FLD_NAME,WHERE_FLD,WHERE_VALUE] **Note:** The field name referenced here should be the source value of the field. However, the SQL should use the hashed value in its filter.	Valid table name and field name are required. The WHERE_FLD and WHERE_VALUE are optional, but if entered, both are required. Use this to only encrypt the field if another field has a certain value. The following is an example. **encrypt=[CI_PERSON,PER_ID_NBR,ID_TYPE_N** The WHERE_VALUE may also

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Description	Valid Values	Comments
			reference another filter. The following is an example. **encrypt=[CI_PERSON,PER_ID_NBR,ID_TYPE_N**

Examples:

- **label=F1_NBR_DAYS type=NUMBER**
- **label=F1_SHOW_ALL_REQ_FLG type=LOOKUP lookup=F1_SHOW_ALL_REQ_FLG**
- Filter value where a Filter UI Map is defined and Description is one of the filters. **type=STRING xpath=description likeable=S**
  - **type=STRING label=DESCR likeable=S divide=below**
  - **label=REQ_TYPE_CD type=TABLE table=F1_REQ_TYPE**

Hidden Filters

Data explorer zones include the ability to define Hidden filters to restrict the rows and / or columns that appear in the zone. The following are the potential sources of a hidden filter's value:

- The global area contains the fields whose values are maintained in global context.
- The portal area contains the fields describing the object currently displayed in a portal.
- Other zones on a portal can broadcast information to the portal area, which can then in turn be used by the zone as a hidden filter.

These parameters are applicable to the zone types

- Info Data Explorer - Multiple SQLs (**F1–DE**)
- Query Data Explorer - Multiple SQLs (**F1–DE-QUERY**)
- Info Data Explorer - Single SQL (**F1–DE-SINGLE**)

A hidden filter is defined using the following mnemonics:

Mnemonic	Description	Valid Values	Comments
name=	This mnemonic defines the name of the field that needs to be broadcast from other zones or populated in the portal context	FIELD_NAME	
datasource=	This mnemonic defines the source of the hidden filter's value. If this mnemonic is left blank, the default behavior is as follows: - If the field has been broadcast from another zone, the broadcast value is used. - If no value is broadcast, the portal context is checked to determine if	**G**	Indicates that the zone should look for the filter value in global context.
		P	Indicates that the zone should look for the filter value in portal context.
		D	Indicates that the zone should look for the filter value in the page data model.

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Description	Valid Values	Comments
	this field exists (if so, its value is taken). - If still no value, the global context is checked. - If still no value, the zone appears as per the poprule mnemonic.		
poprule=	This mnemonic controls what happens if the hidden filter is not present.	**R**	Indicates that a value for the filter is required. The zone will be set to the "empty state" and the "please broadcast" message will appear in the zone. This is the default value.
		O	Indicates that the value is optional. If no value is required, the zone is still built without that value.
type=	Defines the visual metaphor used to capture the filter values.	**DATE**	Filters of this type capture a date.
		DATE/TIME	Filters of this type capture a date and time.
		STRING	Filters of this type capture a string
		MONEY	Filters of this type capture a monetary field. This type of filter must also reference the cur mnemonic.
		NUMBER	Filters of this type capture a numeric field. This type of filter may also reference the decimals mnemonic.
		LOOKUP	Filters of this type capture a lookup value. This type of filter must also reference the lookup mnemonic.
		TABLE	Filters of this type capture an administrative

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Description	Valid Values	Comments
			table's value (code and description). This type of filter must also reference the table mnemonic.
		CHARTYPE	Filters of this type capture predefined characteristic values for a characteristic type (code and description). This type of filter must also reference the chartype mnemonic.
		ASIS	Filters of this type capture a list of values to be referenced within an 'IN' clause within the SQL statement.
label=	Defines the filter's label that appears in the zone's description bar.	FIELD_NAME	Enter a valid field name whose label should be used. This should always be the option used if multiple languages are needed.
		'text'	Defines the text directly.
cur=	Defines the currency code applied when **type=MONEY**.	CURRENCY_CD	Enter a reference to a valid currency code.
dec=	Defines the number of decimal places when **type=NUMBER**.	N	It is optional. If provided it should be an integer. If not provided, the number of decimals will default to the number of decimal places defined on the currency code specified on the installation record.
lookup=	Defines the lookup flag whose values appear when **type=LOOKUP**.	LOOKUP_FIELD_NAME	Enter a reference to a valid lookup field name.
table=	Defines the admin table whose values appear when **type=TABLE**.	TABLE_NAME	Enter a reference to a valid admin table name.

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Description	Valid Values	Comments
chartype=	Defines the characteristic type code whose values appear when **type=CHARTYPE**.	CHAR_TYPE_CD	Enter a reference to a valid characteristic type code.
searchField=	This mnemonic controls the initial population of the filter when the zone is launched as a search from a UI map.	FIELD_NAME	Enter the field name that exactly matches the searchField name specified in the oraSearchField html element in the UI map.

Multi-Select Action

This parameter defines an action to be included in the action area for multi-selection processing. Note that a multi-selection action can only be used if the Multi Select parameter has been set to YES, which causes a checkbox to appear on each row displayed. The action defined here will trigger against all rows selected by the user via the checkbox.

These parameters are applicable to the zone types

- Info Data Explorer - Multiple SQLs (**F1–DE**)
- Query Data Explorer - Multiple SQLs (**F1–DE-QUERY**)
- Info Data Explorer - Single SQL (**F1–DE-SINGLE**)

A multi select action has the following mnemonics:

Mnemonic	Description	Valid Values	Comments
script=	This mnemonic defines the script to be invoked when the action is clicked. This is required.	SCR_CD	Enter a reference to a valid BPA script or Service Script.
type=	This mnemonic defines how the action should be rendered.	**BUTTON**	The action is rendered as a button. This is the default.

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Description	Valid Values	Comments
		LINK	The action is rendered as hypertext.
		ICON	The action is rendered as a graphic icon. For this option, the icon mnemonic is required.
icon=	This mnemonic defines the icon to display when **type=ICON**.	DISPLAY_ICON_CD	Enter a reference to a valid display icon code.
refresh=	This mnemonic indicates how and if a refresh should occur after the script completes.	**NO**	Indicates that no refresh is performed. This is the default.
		ZONE	Indicates that a refresh of the zone is performed.
		PORTAL	Indicates that a refresh of the entire portal is performed.

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Description	Valid Values	Comments
label=	By default, the button label, link text or icon tooltip will come from the script description. Use this mnemonic to override that label.	FIELD_NAME	Enter a valid field name whose label should be used. This should always be the option used if multiple languages are needed.
		'text'	Enter the text directly in single quotes.
list=	When executing the script, the framework builds an XML list containing information from each row selected. This list must be defined in the script's schema and referenced in	listElementName	Enter a valid list element name from the script schema.

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Description	Valid Values	Comments
	this mnemonic.		
context=[elementName1=rowData1 elementName2=rowData2]	This mnemonic is used to populate the list with the appropriate information from each selected row. The mnemonic supports passing multiple values. In each case the element in the schema list is defined first followed by an equal sign, followed by information about the data used to populate the element defined using one	Cx	Indicates that the element should be populated with a value in the referenced column parameter.
		Px	Indicates that the element should be populated with a value in the referenced post processing parameter.
		COLUMN_NAME	Indicates that the element should be populated with a value from a column in the SQL statement.
		'constant'	Indicates that the value defined in single quotes should be passed.

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Description	Valid Values	Comments

	of the valid values defined in the next column.
	
	One or more values may be defined. Each context value is defined separated by spaces. The whole set of context values should be surrounded by square brackets.
	
	Example of a schema:
	
	`<schema>`
	`–`
	`–`
	`<accountInfo –`
	`type="list">`
	`–`
	`–`
	`–`
	`–`
	`<accountId/`
	`>`
	`–`
	`–`
	`–`
	`–`
	`<name/`
	`>`		

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Description	Valid Values	Comments
	– – – – <amount/> – – – – <process/> – – </accountInfo> </schema> Example of list and context mnemonics. **list=accountInfo** **context=[accountId=ACCT_ID name=C2 amount=P3 process='O']**		
class=	Use this mnemonic to override the look and feel of the action using a different CSS style.	'className1' 'className2'	Enter one or more classes in single quotes to be appended to the standard class(es). Multiple class names may be provided.
style=	Use this mnemonic to override the look and	Standard style= format.	All allowed css style definitions may be used.

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Description	Valid Values	Comments
	feel of the action element using the indicated css style.		

Data Explorer Graph Configuration

For some data explorer zones, it is more effective to display the results in a graphical form such as a pie or line chart. Visualization option parameters enable the graph configuration to be integrated with the data explorer zone that retrieves the data.

The visualization option parameters support three options:

- **Grid** - displays the data in a standard data explorer grid
- **Chart** - displays the data in the form of a chart
- **Map** - provides the ability to invoke a UI Map to display the data

Up to five visualization option parameters may be specified. One of the options must be the standard grid, allowing for a maximum of four graphical options. The zone will display icons corresponding to the visualization types to allow you to switch between the various views. These icons are presented in the same order are they are defined in the parameter list.

Note: If the only display option is a standard grid, it is not necessary to specify the grid as a visualization parameter.

Visualization option parameters are provided on the Info Data Explorer - Single SQL (**F1–DE-SINGLE**) and Info Data Explorer - Multiple SQLs (**F1–DE**) zone types.

Visualization Option Parameter Configuration

Mnemonics are used to define the attributes for an option type within the parameter value.

For a **grid**, the mnemonic is ‘optiontype=GRID’. No other mnemonics are required.

For a **map**, the mnemonics are ‘optiontype=MAP map=MAP_CD schemaPath=xpath’, where ‘MAP_CD’ is the ID of the UI Map and ‘xpath’ is the schema path of the UI Map. In most use cases the schema elements will match the result set from the data explorer, in which case no schemaPath mnemonic is needed. If the UI Map schema has additional elements present, the schemaPath attribute will define the group node that the data explorer information will be copied into.

For a **chart**, the option type mnemonics is ‘optiontype=CHART’. This is followed by the appropriate chart configuration attributes, which are described in the section below.

Chart Configuration

The following attributes are used to configure a graphical representation of a set of data.

The ‘**oraChart:**’ attribute defines the type of graph to display and its general configuration. The set of configuration parameters available for this attribute are as follows:

Parameter	Values	Description
type:	**pie** **stacked** **cluster**	Defines the type of chart to display. Required

Copyright © 2024, Oracle and/or its affiliates.

Parameter	Values	Description
	line **area** **combo**	
showLegend:	**true** **false**	Defines if the chart should have a legend displayed. Optional (default is **true**)
legendPosition:	**left** **right** **bottom** **top**	Defines where the legend should appear. Optional (default is **right**) Setting position to **left** or **right** will automatically render it vertically. Setting position to **top** or **bottom** will automatically render it horizontally.
legendBorder:	**true** **false**	Defines if the legend should display with a border around it. Optional (default is **false**)
depth:	**true** **false**	A value of **true** indicates a 3 dimensional depth for the chart. Optional (default **false**, which is a 2 dimensional chart)
animate:	**true** **false**	A value of **true** indicates that the graph should animate when displayed. Optional (default is **true**). When using large data sets, consider disabling animation.
dataCursor:	**on** **off**	A value of **on** enables hovering anywhere in the graph. Optional (default is **off**). It is not applicable to pie charts.
orientation:	**horizontal**	Defines the chart orientation. This only applies to bar, line, area, combo charts. E.g., **oraChart="type:cluster; orientation:horizontal"**, defines horizontal cluster chart. Optional (default is vertical).

The **oraChartSeries** attribute defines the source information for the graph. Most charts support 5 of these attributes, with the following mnemonics:

- **oraChartSeries1**
- **oraChartSeries2**
- **oraChartSeries3**

Copyright © 2024, Oracle and/or its affiliates.

- **oraChartSeries4**
- **oraChartSeries5**

Stacked charts support an unlimited number of series by continuing to add attributes **oraChartSeries6** and above, but beware of performance implications and memory limits when using an excessively high number of series. All attributes are identical in format and accept the same parameters, as described below.

Note: All the charts require **oraChartSeries1**. Stacked, Cluster and Line charts may optionally include the additional series attributes (for multiple bars/lines).

If you define multiple series, then data must be provided for all series defined. The data amounts can be 0 (zero) but they have to be present in order for the chart to display correctly.

The set of configuration parameters available for the **oraChartSeriesN** attribute are:

Parameter	Values	Description
list:	XPath value	Defines the XPath to the list in the XML that contains the data to chart. Required only for HTML definitions in UI Maps. Data explorer values are derived from the columns for the zone.
amount:	Cn Pn XPath value	Defines the element containing the amount to chart. For data explorers, this will be a SQL or post-processing column identifier. For HTML definitions, the XPath of the element must be specified. Required
xaxis:	Cn Pn XPath value	Defines the element containing the x-axis data. For data explorers, this will be a SQL or post-processing column identifier. For HTML definitions, the XPath of the element is specified. Required for Stacked, Cluster, Line, Area and Combo charts.
xaxisFormat:	**date** **dateTime** **time** **localDate** **string**	Defines the x-axis data format. If it is **date**, **dateTime** or **time** then the value is presented in the format as defined on the user's display profile. In case of **localDate** or **string** the data is displayed as is with no special formatting. Optional (Default is **date**).
label:	Text value	Defines the label for the amount being charted. Either this setting or **labelPath:** must be defined.
labelPath:	Cn Pn	Defines the element that provides the label for the amount being charted. For data explorers, this will be a SQL or post-processing column

Copyright © 2024, Oracle and/or its affiliates.

Parameter	Values	Description
	XPath value	identifier. For HTML definitions, the XPath of the element must be specified. Either this setting or **label:** must be defined.
currency:	A valid Currency code	Defines the currency code for the amount being charted. Optional.
currencyPath:	Cn Pn XPath value	Defines the element that provides the currency code for the amount being charted. For data explorers, this will be a SQL or post-processing column identifier. For HTML definitions, the XPath of the element must be specified. Optional.
hoverText:	Text value	Defines the hover text for the chart element. Optional (A default hover text is always available.) Ignored if **hoverTextPath:** is defined. The following substitution variables are available. • **$label** This will be replaced with the label text as determined by the **label:** or **labelPath:** setting. • **$amount** This will be replaced with the amount text as specified by the **amount:** setting. • **$axis** This will be replaced with the x-axis text. • **$%** This will be replaced by the percentage "slice" of the pie or bar. • **$newline** This will force a line break. If the hover text defined contains any of the above values, they will be replaced by the equivalent text prior to being displayed. Example: `"hoverText:$label$newline $amount"`
hoverTextPath:	Cn Pn XPath value	Defines the element that provides the hover text for the chart element. For data explorers, this will be a SQL or post-processing column identifier. For HTML definitions, the XPath of the element must be specified.

Copyright © 2024, Oracle and/or its affiliates.

Parameter	Values	Description
		The hover text in the element can utilize all the substitution functionality described above in the **hoverText:** description. Optional.
type:	**bar** **line** **area**	This attribute is used for the **combo** chart type only. It defines how each series on the **combo** chart should be presented. The following example defines a **combo** chart where one series is rendered as bars and another one as area. `oraChart="type:combo;" - oraChartSeries1="xaxis:C3; - label:C2; - amount:C1;type:bar" - oraChartSeries2="xaxis:C3; - label:C2; - amount:C1;type:area"`
navOpt:	A valid Navigation Option code.	Defines a navigation option to be activated when the chart element is clicked. Optional.
navOptPath:	Cn Pn XPath value	Defines the element that provides the navigation option to be activated when the chart element is clicked. For data explorers, this will be a SQL or post-processing column identifier. For HTML definitions, the XPath of the element must be specified. Optional Note that both **navOpt:** and **navOptPath:** may be configured. The **navOptPath:** navigation option is processed first. If a value is found it is used, otherwise the value in the **navOpt:** setting is used. This means that the chart definition can include a default navigation option and a navigation option present in the data will override it.
key:	Cn Pn XPath value	Defines the element that contains the key field data to be used in a navigation option. For data explorers, this will be a SQL or post-processing column identifier. For HTML definitions, the XPath of the element must be specified. This is required if either **navOpt:** or **navOptPath:** is defined.

Copyright © 2024, Oracle and/or its affiliates.

Parameter	Values	Description
		Note: Only one key field can be configured for a navigation option.
script:	A BPA script name	Defines a BPA script to be activated when the chart element is clicked. Optional When a script is executed, all the elements from the chart data will be made available to the script as temporary variables.
scriptPath:	Cn Pn XPath value	Defines the element that provides the BPA script to be activated when the chart element is clicked. For data explorers, this will be a SQL or post-processing column identifier. For HTML definitions, the XPath of the element must be specified. Optional Note that both **script:** and **scriptPath:** can be configured. The **scriptPath:** script option is processed first. If a value is found it is used, otherwise the value in the **script:** setting is used. This means the chart definition can include a default script and a script present in the data will override it.
color:	HTML Color code / RGB value	Optional (default colors applied) Defines the color for the series. The format is a valid HTML color code, e.g. **green** or **blue**. All valid color names are defined in this link: http://www.w3schools.com/htmL/html_colornames.asp. Alternatively an RGB format may be used. (**00FF00** is green and **0000FF** is blue) **Note:** Refer to Color Contrast for information about the use of the HTML color 'red' and its impact on accessibility.
colorPath:	Cn Pn XPath value	Defines the element containing a color for the series. For data explorers, this will be a SQL or post-processing column identifier. For HTML definitions, the XPath of the element must be specified. The valid formats as described in the **color:** setting apply. Optional (default colors applied)
pieColors:	HTML color code / RGB values	Defines the colors for the pie series. Any number of HTML color codes or RGB color

Copyright © 2024, Oracle and/or its affiliates.

Parameter	Values	Description

		values can be specified, separated by spaces. Examples:
		`pieColors: green blue`
		`pieColors: 00FF00 0000FF`
		Optional (default colors are applied if the series exceeds the values specified)

The **oraChartBroadcast:** attribute provides the ability to broadcast a value from a chart to the portal context. An unlimited number of fields can be broadcast, configured as name/value pairs; for example: oraChartBroadcast="FIELD_NAME1:C1;FIELD_NAME2:C2;" where FIELD_NAMEn is the name of the portal context field to be broadcast and Cn (or Pn) is the element that contains the data value to be broadcast. Broadcast is triggered by clicking on a corresponding section of the chart.

For HTML definitions in UI Maps, the syntax is oraChartBroadcast="FIELD_NAME1:xpath1;FIELD_NAME2:xpath2;" where 'xpathn' is the element in the list schema that contains the data value to be broadcast.

Zone Configuration Examples

Zone with a simple pie chart

In this example, the zone SQL summarizes To Do entries by Role. The column definitions are:

- C1 - role description
- C2 - count of To Do entries by role
- C3 - role ID

Visualization Option 1

This option shows To Do entries by Role as a pie chart. The role ID will be broadcast to the portal when the corresponding section of the pie chart is clicked.

```
optiontype=CHART
oraChart="type:pie;"
oraChartSeries1="labelPath:C1; amount:C2;hoverText:$label$newline$amount;"
oraChartBroadcast="ROLE_ID:C3"
```

Zone with two chart types

In this example, the zone SQL summarizes To Do entries by Priority and also calculates separate counts by To Do status within priority. The column definitions are:

- C1 - priority description
- C2 - count of To Do entries by priority
- C3 - priority flag value
- C4 - count of Open To Dos by priority
- C5 - count of Assigned To Dos by priority
- C6 - Label for the Open To Dos bar in the cluster chart
- C7 - Label for the Assigned To Dos bar in the cluster chart

Visualization Option 1

Copyright © 2024, Oracle and/or its affiliates.

This option shows To Do entries by Priority as a pie chart. The priority value will be broadcast to the portal when the corresponding section of the pie chart is clicked.

```
optiontype=CHART
oraChart="type:pie; legendBorder:true;"
oraChartSeries1="labelPath:C1; amount:C2;hoverText:$label$newline$amount;"
oraChartBroadcast="TD_PRIORITY_FLG:C3"
```

Visualization Option 2

This option shows To Do entries by Priority as a cluster chart with two data series, one for Open To Dos and another for Assigned To Dos . The corresponding priority value will be broadcast to the portal when clicking either the Open or Assigned bar of the cluster chart.

```
optiontype=CHART oraChart="type:cluster;  showLegend:true;" -
oraChartSeries1="xaxis:C1; xaxisFormat:string; labelPath:C6; -
amount:C4;hoverText:$label$newline$amount;"
oraChartSeries2="xaxis:C1; xaxisFormat:string; labelPath:C7; -
amount:C5;hoverText:$label$newline$amount;"
oraChartBroadcast="TD_PRIORITY_FLG:C3"
```

Pagination Configuration

The various data explorer zones in the product support the ability to configure pagination so that a user can 'page through' a large set of results using "previous" and "next" buttons or links.

There are several zone parameters that are impacted when attempting to configure this functionality. The following steps highlight the configuration.

- The **Enable Pagination** parameter must be configured to define the basic setup for pagination for the zone. This parameter defines whether the "previous" and "next" actions are defined as buttons, links or icons and indicates the location of the actions. It also allows an indication as to whether the additional rows are simply appended, rather than shown in a new "page". Refer to the parameter's embedded help for information about the specific syntax.
- It is recommended that the zone is configured with record count and page information by properly configuring the **Record Count Display** parameter. Refer to the parameter's embedded help for information about the specific syntax.
- Configure the **Number of Rows to Retrieve for SQL**. parameter to define the number of records displayed per page. If this parameter is not specified the value in the **Number of Rows to Display** parameter is used.
- Configure the key that will be used for paging so that the system can keep track of the 'page break'. The data must be sorted by the paging key; as a result, the decision for identifying the paging key must take into account the design for the zone and the data being displayed. In addition, the paging key must be unique to ensure that the page breaks occur correctly. See below for configuration examples.
  - The **SQL Statement** must includes additional clauses **PAGENEXT** and **PAGEPREV** based on the paging key. In addition, as mentioned above, the paging key must be used in the ORDER BY clause.
  - The SQL **Column** parameters must define the paging key mnemonic to be used in conjunction with the SQL statement paging clauses.
  - It is recommended to configure the **SQL Display Column** parameter to show the data in the same order as the ORDER BY clause.

The following zone types support this capability:

- Info Data Explorer - Single SQL (**F1–DE-SINGLE**).
- Info Data Explorer - Multiple SQLs (**F1–DE**). Note that zones of this type support a union of the results of all the SQL statements. As a result, pagination may only be enabled for zones of this type if a single SQL is used. The system is not able to keep track of the pagination across disparate SQL statements.
- Query Data Explorer - Multiple SQLs (**F1–DE-QUERY**).
- Multi Query Data Explorer (**F1–DE-MULQRY**). Zones of this type do not include configuration for SQL statements or column display. However, they do include configuration for the **Enable Pagination**. This parameter must be configured in order for pagination on the individual zones to work.

Copyright © 2024, Oracle and/or its affiliates.

> **Note:**
>
> **Zones used for a Business Service.** Note that pagination is ignored when invoking a data explorer zone via a business service. In this scenario, the zone will return the first "chunk" of rows as defined by the Number of Rows parameters.
>
> **Zones with data visualization options.** Pagination is not supported for zones that define visualization options to show data in a graphical form.

Examples

Simple Paging Key

In this example, the Extendable Lookup Value is defined as Column 1 (C1) and is marked as the paging key. This field is unique for the table and works well as a simple paging key.

```
SELECT A.F1_EXT_LOOKUP_VALUE,A.BUS_OBJ_CD
FROM -
  F1_EXT_LOOKUP_VAL A, -
  F1_EXT_LOOKUP_VAL_L B
WHERE -
A.BUS_OBJ_CD = -:H1
AND A.BUS_OBJ_CD = B.BUS_OBJ_CD
AND A.F1_EXT_LOOKUP_VALUE = B.F1_EXT_LOOKUP_VALUE
AND B.LANGUAGE_CD = -:LANGUAGE
[(F1) AND UPPER(A.F1_EXT_LOOKUP_VALUE) like UPPER(:F1)]
[(F2) AND ((UPPER(B.DESCR_OVRD) like UPPER(:F2))
OR (B.DESCR_OVRD = -' -' AND UPPER(B.DESCR) like UPPER(:F2)))]
[(PAGENEXT) AND A.F1_EXT_LOOKUP_VALUE > -:C1]
[(PAGEPREV) AND A.F1_EXT_LOOKUP_VALUE < -:C1]
ORDER BY A.F1_EXT_LOOKUP_VALUE
```

Complex Paging Key

Most queries however do not sort by a unique value. In this case, the paging key needs to be set based on the sorting of the query and should include a unique field, such as the primary key, as the last paging key. In this example, the query is showing results sorted by To Do Type, Role and User. All fields, including the To Do Entry ID (the primary key) are marked as paging keys.

```
SELECT TD_TYPE_CD, ROLE_ID, ASSIGNED_TO, ASSIGNED_DTTM, TD_PRIORITY_FLG, -
TD_ENTRY_ID
FROM  CI_TD_ENTRY
WHERE -
ENTRY_STATUS_FLG IN ('O', -'W')
[(F1) and TD_TYPE_CD = -:F1]
[(F2) AND ASSIGNED_TO = -:F2]
[(F3) AND ROLE_ID = -:F3]
[(PAGENEXT) and ((TD_TYPE_CD>:C1) or (TD_TYPE_CD=:C1 and ROLE_ID>:C2) or -
(TD_TYPE_CD=:C1 and ROLE_ID=:C2 -
and ASSIGNED_TO>:C3) or (TD_TYPE_CD=:C1 and ROLE_ID=:C2 and -
ASSIGNED_TO=:C3 AND TD_ENTRY_ID>:C4))] -
[(PAGEPREV) and ((TD_TYPE_CD<:C1) or (TD_TYPE_CD=:C1 and ROLE_ID<:C2) or -
(TD_TYPE_CD=:C1 and ROLE_ID=:C2 -
and ASSIGNED_TO<:C3) or (TD_TYPE_CD=:C1 and ROLE_ID=:C2 and -
ASSIGNED_TO=:C3 AND TD_ENTRY_ID<:C4))] -
ORDER BY TD_TYPE_CD, ROLE_ID, ASSIGNED_TO, TD_ENTRY_ID
```

Use Data Explorer for Derived Data

There are times when a design warrants displaying data in a data explorer zone that is not accessible via SQL. For example, perhaps the data is from another system and it requires a web service call. The JMS Message Browser is another example.

Copyright © 2024, Oracle and/or its affiliates.

The product provides functionality in the data explorer that allows you to call a script after the user filters are populated but before the SQL is executed. The script can retrieve the data as appropriate, store the data in table format so that the SQL can retrieve the data from the table.

The following points provide more detail:

- Create a service script that retrieves the data as needed. This script should store the retrieved data in a temporary table.
  - The product provides a table that may be used. It is called F1_GENERIC_GTT (Generic Global Temporary Table). There is a business service - Create Global Temporary Table Records (**F1-InsertGTTRecords**) that the service script may call to insert the records.
  - Note that if the data is accessed via a web service call, it may be appropriate to execute the web service in a separate session using the business service F1-ExecuteScriptInNewSession to trap errors that may be issued by the web service call and provide a better error.
- In the data explorer zone use this service script in the zone's pre-processing script parameter. If any user or hidden filters should be passed into the script, the parameter supports mnemonics for this purpose. Refer to the parameter's embedded help for the supported syntax.
- The SQL for the data explorer should access the temporary table that was populated by the service script.

Configuring Timeline Zones

This topic highlights information related to configuring a timeline zone. The zone type is **F1-TIMELINE**. A timeline zone contains one or more "lines" where each line shows when significant events have occurred. The output of each line is driven by an algorithm configured on a timeline zone. Each algorithm is responsible for retrieving a single type of data. For example, one algorithm may retrieve bills for an account in a given time period whereas another algorithm may retrieve payments for that account for the same time period.

The algorithms to configure for a timeline zone use the **Zone - Timeline** plug-in spot. Please note the following details about the behavior for algorithms for this plug-in spot.

- The timeline algorithm receives all the global context values currently populated. In addition, it receives a start and end date from the zone, based on the time period chosen by the user, along with the maximum number of events that can be reasonably displayed for the chosen time period. The algorithm should use this information to retrieve data for a given type of transaction related to one or more of the input context values for the provided time period.
- For each event found, the algorithm returns information about the event along with many options that assist the user in getting more detail about each event or acting on an event.
  - Event date
  - Primary key of the record (key / value pairs)
  - FK Reference. With this information, the timeline zone will display the appropriate info string to display in the zone's info area when clicking on the event. In addition, the FK reference identifies the appropriate navigation option to use when a user clicks the info string hypertext to view the record on its maintenance page.
  - Background Color and Text Color to use for the event. (Obsolete). This information is no longer used by the timeline zone and will be ignored.
  - Icon use for the event. (Obsolete). This information is no longer used by the timeline zone and will be ignored.
  - BPA script to launch when a user clicks on an event. (Optional). The algorithm may return one or more BPA scripts that a user may launch to act on an event. For example, for an event that has a status of **Error**, perhaps a BPA is provided to walk a user through resolving the error.

    When a script is initiated from a timeline, the system puts the prime key of the event into a field in the page data model. The name of the field is the column name(s) of the event's prime key. For example, when a script associated with a payment event is kicked off, the system populates a field called PAY_ID with the prime-key of the selected payment.

Note that your specific edge application may supply algorithm types for a timeline zone as part of the base product. Although algorithm types may be provided, typically the product does not deliver algorithms because the parameters for the algorithms are driven by a particular implementation's business rules and preferences. As a result, the product will also not deliver pre-configured timeline zones. Please refer to your edge application's documentation for more information about what timeline algorithm types are delivered, if any and recommendations for configuration.

Copyright © 2024, Oracle and/or its affiliates.

Defining Context-Sensitive Zones

A context-sensitive zone allows you to associate a zone with a specific user-interface transaction. A context-sensitive zone appears at the top of the Sidebar when a user accesses a page for which the zone is specified as the context. For example, when viewing a business object, additional zones are visible that are specific to the business object page.

CAUTION: Make sure that the zone is appropriate for the transaction on which you are specifying it. For example, if your zone requires a business object ID as one of its keys, it should not be displayed on the To Do entry transaction.

Select **Admin** > **Context Sensitive Zone** to maintain context-sensitive zones.

Description of Page

The **Navigation Key** is a unique identifier of a tab page within the system. **Owner** indicates if this navigation key is owned by the base product or by your implementation (**Customer Modification**).

CAUTION: Important! When introducing a new context sensitive zone, carefully consider its naming convention. Refer to System Data Naming Convention for more information.

The grid contains the list of context-sensitive zones and the sequence in which they appear in the sidebar for the selected navigation key. The grid contains the following fields:

- **Zone** is the name of the zone to display in the Sidebar.
- **Sequence** is the sequence in which the zone is displayed (if multiple context-sensitive zones are defined).
- **Owner** indicates if this context sensitive zone is owned by the base product or by your implementation (**Customer Modification**).

Where Used

A context-sensitive zone displays at the top of the Sidebar whenever a user accesses the transaction for with the zone is specified.

Defining Portals

This portal is used to view and maintain the configuration of portals. An implementation may define their own portals or may override some of the settings for base product provided portals.

You can access the portal from the **Admin Menu** > **System** > **Portal**. You are brought to a query portal with options for searching for a specific record. Once a portal record has been selected you are brought to the maintenance portal to view and maintain the selected record.

Note: When introducing a new record, carefully consider its naming convention. Refer to System Data Naming Convention for more information.

Refer to Understanding Portals and Zones for more information.

The following zones may appear as part of the portal's **Main** tab page

- **Portal.** This zone provides information about the selected portal definition.
- **Options.** This zone lists various configuration options for the selected portal record.
- **Tabs.** This zone lists other tab portals if any are configured for a standalone portal. This zone is only visible when such tabs exist for the current standalone portal.
- **Context Sensitive Zones.** This zone lists context sensitive zones that appear on the sidebar when this portal is displayed . This zone is only visible when such zones exist for the current standalone portal.

Copyright © 2024, Oracle and/or its affiliates.

Defining Menu Options

This portal is used to define / change any menu in the system. You can access the portal from the **Admin Menu** > **System** > **Menu**. You are brought to a query portal with options for searching for a specific menu. Once a menu has been selected you are brought to the maintenance portal to view and maintain the selected record.

CAUTION: Updating menus requires technical knowledge of the system. This is an implementation and delivery issue and should not be attempted if you do not have previous experience with menus.

Note: Security and menus. Refer to Secured Objects and Application Services for a discussion of how application security can prevent menu items (or an entire menu) from appearing.

Note: Module configuration and menus. Your module configuration can prevent menu items (or an entire menu) from appearing.

The following zones may appear as part of the portal's **Main** tab page:

- **Menu.** This zone provides information about the selected menu.
- **Menu Lines.** This zone lists the menu's lines and supports actions to add, edit and delete lines. You may also broadcast a line to review more information about its items configuration.
- **Menu Line Items.** This zone lists item information for a selected menu line.

Application security information for the menu can be reviewed and managed from the **Application Security** tab. The following zones may appear as part of this tab page:

- **Menu Lines.** This zone lists the menu's lines. Broadcast a line to review its application security information.
- **Menu Line Security.** This zone lists application services related to the selected menu line. You may use this zone to review and configure security access for the selected menu line and a specific user group.
- **Menu Line Security Details.** This zone lists the entities secured by a selected application service.

The following zone may appear as part of the portal's **References** tab page:

- **Menu References.** This zone lists entities that reference the selected menu.

The following sections provides more information.

Menu Types

The following types of menus are supported:

- **Admin** is one of the menus that appears in the Application Toolbar. It is a special type of menu because admin menu items can be grouped alphabetically or by functional group. Refer to the description of Admin Menu Order on Installation Options - Framework for more information about admin menu options.
- **Main** is another menu that appears in the Application Toolbar that is simply titled *Menu*.
- **Submenu** defines a menu group that appears when an Application Toolbar menu is selected. for the Admin menu, this is only visible when it's organized functionally.
- **Context** refers to a context menu.
- **Page Action Menu** defines buttons that appear in the Page Title Area.
- **User Menu** refers to the menu items that appear on the user menu; for example, User Preferences.

Extending Product Owned Menus

An implementation may choose to add custom menu lines along with its menu item (or items) to a base product owned menu.

Copyright © 2024, Oracle and/or its affiliates.

Overriding Menu Lines Sequence

An implementation may override the sequence of a base product owned menu line. Also note that the sequence is defined on the menu line language table, allowing for different orders to be used for different languages (or to let the menu be sorted alphabetically in one language and in a specified order in a different one).

Menu Information is Cached

Use the **Flush Menu** button to flush the cached menu items so you can see any modified or newly created menus. Refer to Caching Overview for more information.

Defining Navigation Keys

Each location to which a user can navigate (e.g., transactions, tab pages, tab menus, online help links, etc.) is identified by a navigation key. A navigation key is a logical identifier for a URL.

Navigation Key Types

There are three types of navigation keys:

- **System navigation keys** define locations where the target for the navigation is a transaction or portal within the system. The navigation keys define the program component that identifies the page to navigate to.
- **External navigation keys** define a URL that identifies the target location. External URLs can be specified as relative to the product web server or fully qualified. External navigation keys always launch in a new instance of a browser window. Examples of external navigation keys include application viewer links and URLs to external systems.
- **Help navigation keys** define a online help topic that identifies the specific page within the online help to launch. Help navigation keys may be related to a program component when the help is related to a specific page in system.

Navigation Key vs Navigation Option

The system has two entities that work in conjunction with each other to specify how navigation works:

- Navigation Key defines a unique location to which a user can navigate. For example, each page in the system has a unique navigation key. Navigation keys can also define locations that are "outside" of the system. For example, you can create a navigation key that references an external URL. Think of a navigation key as defining "where to go".
- Navigation Option defines how a page is opened when a user wants to navigate someplace. For example, you might have a navigation key that identifies a specific page. This navigation key can then be referenced on two navigation options; the first navigation option may allow users to navigate to the page with no context included, while the second navigates to the page with context data provided to automatically display information related to that context.
- Please note that a wide variety of options can be defined on a navigation option. In addition to defining if data is passed to the page, it could also define search options. In addition, there are some navigation options that do not reference a navigation key but rather refer to a BPA script that should be launched.

The Flexibility of Navigation Keys

Navigation keys provide a great deal of functionality to your users. Use navigation keys to:

- Allow users to navigate to new pages or search programs
- Allow users to transfer to an external system or web page. After setting up this data, your users may be able to access this external URL from a menu, a context menu, their favorite links, etc. Refer to Linking to External Locations for more information.

Refer to the Tool Suite Guide for more information on developing program components.

Copyright © 2024, Oracle and/or its affiliates.

> **Note: Replacing Base-Package Pages or Searches**. If your new page or search has been designed to replace a module in the base-package, the navigation key must indicate that it is overriding an existing navigation key.

Linking to External Locations

If you want to include links to external systems or locations from within the system, you need to:

- Define a navigation key that specifies the URL of the location. For example, define an external navigation key that as a URL of **http://www.oracle.com/**.
- Define a navigation option that specifies from where in the system a user can go to your external location. For example, define a navigation option with a usage of **Favorites** or with a usage of **Menu**. Your navigation option points to the navigation key you defined above.
- Add your navigation option to the appropriate location within the system. For example, have users add the navigation option to their Favorite Links or add the navigation option as an item on a menu.

Overriding Navigation Keys

Your implementation may choose to design a program component (e.g., a maintenance transaction or search page) to replace a component provided by the system. When doing this, the new navigation key must indicate that it is overriding the system navigation key. As a result, any menu entry or navigation options that reference this overridden navigation key automatically navigates to the custom component.

For example, if you have a custom On-line Batch Submission page and would like users to use this page rather than the one provided by the system, setting up an override navigation key ensures that if a user chooses to navigate to the On-line Batch Submission from Menu or a context menu, the user is brought to the custom On-line Batch Submission page.

To create an override navigation key, you need to:

- Define a navigation key using an appropriate naming convention.
- If the Navigation Key Type of the navigation key being overridden is **External**, specify a Navigation Key Type of **Override (Other)** and define the appropriate URL Component.
- If the Navigation Key Type of navigation key being overridden is **System**, specify a Navigation Key Type of **Override (System)** and populate the Program Component ID with your custom program component ID.
- Specify the navigation key that you are overriding in the **Overridden Navigation Key** field.

Refer to the Tool Suite Guide for more information about developing your own program components.

Maintaining Navigation Keys

Select **Admin** > **System** > **Navigation Key** to maintain navigation keys.

Use the **Navigation Key Query** portal to search for an existing navigation key. Once a navigation key is selected, you are brought to the maintenance portal to view and maintain the selected record.

> **CAUTION:** Important! When introducing a new navigation key, carefully consider its naming convention. Refer to System Data Naming Convention for more information.

Description of Page

The **Navigation Key** is a unique name of the navigation key for internal use.

Owner indicates if this navigation key is owned by the base product or by your implementation (**Customer Modification**). This information is display-only.

Navigation Key Type includes the following possible values:

- **External** indicates that the location is specified in the **URL Component** field.

Copyright © 2024, Oracle and/or its affiliates.

- **Help** indicates that the navigation key is used to launch online help where the specific help topic is defined in the **URL Component** field.
- **Override (Other)** indicates that the navigation key overrides another navigation key of type **External** or **Help**. For this option, the name of the navigation key being overridden is populated in the **Overridden Navigation Key** field.
- **Override (System)** indicates that the navigation key overrides a system navigation key. For this option, the name of the navigation key being overridden is populated in the **Overridden Navigation Key** field.
- **System** indicates that the navigation key refers to a transaction in the system identified by its program component.

Fastpath: Refer to Navigation Key Types for more information about navigation key types.

Fastpath: Refer to Overriding Navigation Keys for more information about settings required to override a system navigation key.

Program Component ID is the name of the program component identified by the key (for system navigation keys). The program component ID can also be used to specify the transaction with which an online help link is associated.

Overridden Navigation Key is the name of the navigation key that the current navigation key is overriding (if **Override (Other)** or **Override (System)** is selected for the **Navigation Key Type**). Refer to Overriding Navigation Keys for more information.

URL Component is the specific URL or portion of the URL for the navigation key (external and help navigation keys only). The URL can be relative to the product web server or fully qualified.

Open Window Options allows you to specify options (e.g., width and height) for opening a browser window for an external navigation key. (External navigation keys always launch in a new browser window.) You can use any valid features available in the Window.open( ) JavaScript method. The string should be formatted the same way that it would be for the features argument (e.g., **height=600,width=800,resizeable=yes,scrollbars=yes,toolbar=no**). Refer to a JavaScript reference book for a complete list of available features.

Application Service is the application service that is used to secure access to transactions associated with **External** navigation keys. If a user has access to the specified application service, the user can navigate to the URL defined on the navigation key. Refer to The Big Picture of Application Security for more information.

The grid displays menu items that reference the navigation key (actually, it shows menu items that reference navigations options that, in turn, reference the navigation key).

Defining Navigation Options

Every time a user navigates to a transaction, the system retrieves a navigation option to determine which transaction should open. For example:

- A navigation option is associated with every menu item. When a user selects a menu item, the system retrieves the related navigation option to determine which transaction to open.
- A navigation option is associated with every favorite link. When a user selects a favorite link, the system retrieves the related navigation option to determine which transaction to open.
- A navigation option is associated with every node in the various trees. When a user clicks a node in a tree, the system retrieves the related navigation option to determine which transaction to open.
- Etc.

Many navigation options are shipped with the base product and cannot be modified as these options support core functionality. As part of your implementation, you may add additional navigation options to support your specific business processes.

Navigation options may also be used to launch a BPA script.

The topics in this section describe how to maintain navigation options.

Copyright © 2024, Oracle and/or its affiliates.

CAUTION: In order to improve response times, navigation options are cached the first time they are used after a web server is started. If you change a navigation option and you don't want to wait for the cache to rebuild, you must clear the cached information so it will be immediately rebuilt using current information. A special button has been provided on the Main tab of the navigation option transaction that performs this function. Please refer to Caching Overview for information on the various caches.

Navigation Option - Main

Select **Admin** > **System** > **Navigation Option** to maintain a navigation option.

Use the **Navigation Option Query** portal to search for an existing navigation option. Once a navigation option is selected, you are brought to the maintenance portal to view and maintain the selected record.

Description of Page

Enter a unique **Navigation Option** code and **Description**.

CAUTION: When introducing a new navigation option, carefully consider its naming convention. Refer to System Data Naming Convention for more information.

The **Flush System Login Info** button is used to flush the cached navigation options so you can use any modified navigation options. Refer to Caching Overview for more information.

Owner indicates if this navigation option is owned by the base product or by your implementation (**Customer Modification**). This field is display-only. The system sets the owner to **Customer Modification** when you add a navigation option.

Note: You may not change navigation options that are owned by the base product.

Use **Navigation Option Type** to define if the navigation option navigates to a **Transaction**, launches a BPA **Script** or opens an **Attachment**.

Note: The **Attachment** option type is only applicable to navigation options that are used to launch a file attached to a record in the Attachment maintenance object.

For navigation option types of **script**, indicate the **Script** to launch. You can use the **Context Fields** at the bottom of the page if you want to transfer the contents of specific fields to temporary storage variables available to the script. The script engine creates temporary storage variables with names that match the Context Field names.

For navigation option types of **transaction**, define the **Target Transaction** (navigation key) and optionally a specific **Tab Page** (also a navigation key) if a specific tab on the transaction (other than the Main tab) should be opened when navigating to that transaction.

Note: **Finding transaction navigation keys**. When populating the **Target Transaction** and **Tab Page** you are populating an appropriate navigation key. Because the system has a large number of transactions, we recommend using the "%" metaphor when you search for the transaction identifier. For example, if you want to find the currency maintenance transaction, enter "%currency" in the search criteria.

The additional information depends on whether the target transaction is a fixed page or a portal-based page:

- For portal-based pages:
  - **Navigation Mode** is not applicable and should just be set to **Add Mode**.
  - If navigating to a query portal, by default the query portal will open with the default search option defined. If the navigation should open a different search option, define the **Multi-Query Zone** for that query portal and indicate the **Sub-Query Zone** to open by default. Note that for this configuration, it is common to define **Context Fields** to pre-populate search criteria in the target query zone. When using this configuration, be sure that the target query zone's user filters are defined to populate data from context.
- For fixed pages:

Copyright © 2024, Oracle and/or its affiliates.

- **Navigation Mode** indicates if the **Target Transaction** should be opened in **Add Mode** or **Change Mode**.
- **Add Mode** should be used if the option is used to navigate to a transaction ready to add a new object. You can use the **Context Fields** at the bottom of the page if you want to transfer the contents of specific fields to the transaction when it opens.
- **Change Mode** is only applicable for fixed pages and should be used if the option is used to navigate to a transaction ready to update an object. You have two ways to define the object to be changed:
  - Define the name of the fields that make up the unique identifier of the object in the **Context Fields** (and make sure to turn on **Key Field** for each such field).
  - Define the **Search Transaction** (navigation key) if you want to open a search window to retrieve an object before the target transaction opens. Select the appropriate **Search Type** to define which search method should be used. The options in the drop down correspond with the sections in the search (where **Main** is the first section, **Alternate** is the 2nd section, **Alternate 2** is the 3rd section, etc.). You should execute the search window in order to determine what each section does.

    When you select a **Search Type**, define appropriate **Context Fields** so that the system will try to pre-populate the search transaction with these field values when the search first opens. Keep in mind that if a search is populated with field values the search is automatically triggered and, if only one object is found that matches the search criteria, it is selected and the search window closes.

- **Search Group** is only visible if the **Development Tools** module is not turned off. It is used to define the correlation between fields on the search page and the tab page. You can view a tab page's **Search Groups** by viewing the HTML source and scanning for **allFieldPairs**.

The **Go To Tooltip** is used to specify the label associated with the tool tip that appears when hovering over a **Go To** object. Refer to the **Usage** grid below.

The **Usage** grid defines the objects on which this navigation option is used:

- Choose **Favorites** if the navigation option can be used as a favorite link.
- Choose **Menus** if the navigation option can be used as a user's home page or as a menu or context menu item.
- Choose **Script** if the navigation option can be used in a script.
- Choose **Foreign Key** if the navigation option can be used as a foreign key reference.
- Choose **Go To** if the navigation option can be used as a "go to" destination ("go to" destinations are used on Go To buttons, tree nodes, and hyperlinks).
- If your product supports marketing campaigns, you can choose **Campaign** if the navigation option can be used as a "post completion" transaction on a campaign. For more information refer to that product's documentation for campaigns.

The **Context Fields** grid contains the names of the fields whose contents will be passed to the **Target Transaction** or **Script** or used to launch an **Attachment**. The system retrieves the values of these fields from the "current" page and transfers them to the target transaction or to the script's temporary storage. Turn on **Key Field** for each context field that makes up the unique identifier when navigating to a transaction in **Change Mode**.

Note: For an **Attachment**, the grid should contain the Attachment ID.

Note: Navigating from a Menu. The standard followed for many base main **menu** navigation options for fixed transactions is that navigation options launched from the Menu dropdown are configured with no context.

Navigation Option - Tree

This page contains a tree that shows how a navigation option is used. Select **Admin** > **System** > **Navigation Option** and navigate to the **Tree** tab to view this page.

Description of Page

The tree shows every menu item, favorite link, and tree node that references the navigation option. This information is provided to make you aware of the ramifications of changing a navigation option.

Copyright © 2024, Oracle and/or its affiliates.

Understanding System Messages

All error, warning and informational messages that are displayed in the system are maintained on the message table. Every message is identified by a combination of two fields:

- **Message category number.** Think of a message category as a library of messages related to a given functional area. For example, there is a message category for billing messages and another one for payment messages.
- **Message number**. A unique number identifies each message within a category.

Every message has two components: a brief text message and a long description.

Note: You cannot change the product's text. If the message is "owned" by the product, you cannot change the product's message or detailed description. If you want your users to see a different message or detailed description other than that supplied by the product, enter your desired verbiage in the corresponding override fields (and flush the cache).

Defining System Messages

This portal is used to search for message categories and messages that appear throughout the system. An implementation may introduce messages used in custom processes or may choose to override the text for messages delivered by the product.

You can access the portal from the**Admin Menu** > **System** > **Message**. You are brought to a query portal with options for searching for a specific message category or a message. Once a record has been selected you are brought to the maintenance portal to view and maintain the selected record.

Refer to Understanding System Messages for more information.

Defining Message Categories

This portal is used to view and maintain a message category and its messages.

CAUTION: Message category 80000 or greater must be used to define new messages introduced for a specific implementation of the system. Changes to other Message Text will be overwritten when you next upgrade. If you want to make a change to a Message, drill down on the message and populate the Override Message Text. Note that even for message categories 80000 and higher, message numbers lower than 1000 are reserved for common base product messages.

You can access the portal from the**Admin Menu** > **System** > **Message**. You are brought to a query portal with options for searching for a specific message category or a specific message. Once a message category has been selected you are brought to the maintenance portal to view and maintain the selected record.

Refer to Understanding System Messages for more information.

The following zones may appear as part of the portal's **Main** tab page

- **Message Category.** This zone provides information about the selected message category.
- **Messages.** This zone lists and maintains messages for the selected category.

Defining Messages

This portal is used to view and maintain a specific message. An implementation may introduce messages used in custom processes or may choose to override the text for messages delivered by the product.

Copyright © 2024, Oracle and/or its affiliates.

> **CAUTION:** Message category 80000 or greater must be used to define new messages introduced for a specific implementation of the system. Changes to other Message Text will be overwritten when you next upgrade. If you want to make a change to a Message, drill down on the message and populate the Override Message Text. Note that even for message categories 80000 and higher, message numbers lower than 1000 are reserved for common base product messages.

You can access the portal from the**Admin Menu** > **System** > **Message**. You are brought to a query portal with options for searching for a specific message category or a specific message. Once a message has been selected you are brought to the maintenance portal to view and maintain the selected record.

Refer to Understanding System Messages for more information.

The following zones may appear as part of the portal's **Main** tab page

- **Message.** This zone provides information about the selected message.

Understanding Unified Search

Depending on your product, the toolbar search box may support a product specific search for business entities in addition to the standard search for menu items. For example, the search may support a query of business entities by their name, address, contact details etc. The unified search feature is a simplified version of your product's main search, that allows the user to lookup up records using free form search text or hint codes. For more complex queries, the user may navigate to the corresponding advanced search portal.

The configuration of a unified search option includes the following items:

- A **search service script** that implements the data query and defines the supported **filters**. The application service associated with this script is used to secure the unified search option.
- **Search filter rules** and **hints** for the supported filters. These rules are used to parse the text entered by the user to determine whether it qualifies as a valid filter value.
- The **advanced search** portal to use for more complex queries.

The unified search box interacts with a unified search option implementation as follows:

- As the user types free form text in the search box, the application uses the **search filter rules** and **hints** to parse the entered text and translate it to corresponding query filters.
- The **search service script** is then called to query for entities that match these filters and present the results.
- Selecting a result record navigates to the corresponding portal.
- Selecting the **Advanced Search** link in the search box navigates to the general query portal associated with the unified search option.

If a unified search option is not enabled for your product or the user does not have security access to it then the search box may only be used for searching the menu.

The following sections describe various aspects related to the configuration of a unified search option.

The Search Script Controls Everything

A unified search is implemented as a single service script that controls the following:

- **Filters.** The script determines which filters are supported and how they are applied to return results efficiently. Each filter name must be a valid **Field** record.
- **Results.** The script returns a list of business entities that matches the query filters provided by the user, along with the navigation information needed to drill in to a selected record in the list.
- **Application security.** Only users that have access to the script's application service may use the unified search associated with the search script.
- **Override Advance Search.** The script may override the advanced search portal configured for the search option.

Copyright © 2024, Oracle and/or its affiliates.

When the search box component initializes, the script is called once in **Prepare Search Details** mode to return the list of supported filter names. In this mode the script does not perform any query.

As the user types in text in the search box, the search box component analyzes the entered text to derive query filters and calls the search script in **Perform Search** mode to return matching results.

The search box component is designed to support any type of unified search implementation and as such, the interaction between the user interface component and the search script is generic. Filters are defined as name / value pairs and results for each matched entity includes details like primary, secondary and tertiary description lines, along with a line that describes the matching text for entry.

Note: If your organization needs to implement a custom unified search script, it is strongly recommended that you follow the base product's implementation of such scripts as a reference.

Filter Rules and Hints

A unified search is designed to support a predefined set of filters for querying data. For example, searching customers by their address, name, identifiers; or searching locations by address or name etc. Each filter is uniquely identified by a **Field** name as declared by the search service script.

As the user types in free text in the search box, the application applies parsing rules to determine whether the text may be used as the value of one or more filters supported by the unified search. For example, if the user types in the text "ave", parsing rules may determine that it can be used to filter results by a person's name or address but not their email address.

Each parsing rule, defined in the **Search Filter Rule** extendable lookup, associates a filter's field name with a regular expression. Multiple filter rules defined for the same filter name are processed in execution order sequence and it is enough for one rule to return a match for the search text to be used as the value for the specific filter.

Alternatively, the user may use an explicit hint code to identify the filter they would like to use followed by the filter value. For example, an address filter may be associated with the hint "ad:" allowing the user to enter "ad: main" to look up records by address.

Each hint, defined in the **Search Filter Hint** extendable lookup, associates a filter's field name with a hint code.

Note that filter rules and hints are defined for a specific filter name and not a specific unified search option. The same parsing rules may be shared across searches using the same filter names. For example the parsing rule to identify an "email" filter or the "ad:" hint for an address filter are the same regardless of how these are used by a search query. These rules are automatically associated with any unified search option that uses the same filter names.

Note: You may customize the parsing rules and hints associated with a filter by inactivating the base product's extendable lookup records and introducing custom rules that match your business requirements.

Which Unified Search Option is Used?

Each unified search option is defined in the **Unified Search** extendable lookup, referencing the option's search service script and its corresponding advanced search portal. The unified search option is only enabled to users that have access to the search script's application service. When more than one unified search option is configured and enabled for a user, the application chooses the option with the highest sequence as the default option, allowing users to switch to other options they have access to.

Customizing Unified Search

You may customize the parsing rules and hints associated with base product search filters by inactivating the base product's extendable lookup records and introducing custom rules that match your business requirements.

If you need to support a search for different business entities and filters, you need to implement a custom unified search. The configuration of a unified search option involves the following high level steps:

- Design and implement a search script that supports a predefined set of filters and returns the topmost set of matching results while taking into account performance considerations.

Copyright © 2024, Oracle and/or its affiliates.

- Design hints and filter rules as needed for each of the supported filters. Review existing hints and filter rules associated with same filter names when applicable.
- Register each hint, if any in the **Search Filter Hint** extendable lookup.
- Register each filter rule, if any in the **Search Filter Rule** extendable lookup.
- Register the search script in the **Unified Search** extendable lookup.

Trees

Many edge products and implementations make extensive use of UI tree structures to display hierarchical relationships between records in the system. The framework provides an objects to define a tree structure and the rules on how to populate the nodes in the tree, and processes to interpret the configuration and dynamically render the tree display.

The topics in this section provide information about what is supplied in the framework. Your product may supply UI components with ready built trees for common use cases. Refer to your product documentation for more information.

Common Attributes of Trees

Your organization may have many portals and zones showing data in a hierarchical tree. Most trees in the system follow a similar pattern:

- Trees have a number of levels. There may be different types of tree nodes present in any given level; for instance a list of algorithm types for a script, followed by a list of script steps. Icons may be displayed to further identify the node’s object type.
- The most common node types reference an entity within the system such as a schema, script, data area and so on. These nodes usually display the information string from the entity's FK reference and provide the ability to navigate to the entity. They may also allow the node context to be broadcast to other zones in the portal.
- In addition to the ability to navigate to the associated entity, many node types provide a context menu with further navigation paths.
- Some nodes are simply informational and are used to group a set of child nodes below a type or category. For example, the business object Options tree groups the option value nodes below summary nodes that describe the option type.

Each tree defines the nodes that make up the tree and the parent/child relationship between those nodes. A tree node defines which of the common attributes are applicable to the node and the algorithms responsible for retrieving the node data and, optionally, determining the node information to be displayed.

Populate Node Algorithm

A tree node displays a combination of the attributes listed above, typically for an object in context. For a root node, the object will normally be derived from a global or portal context value, such as the current script. For each lower level node, the context is derived from the context fields of its parent row; for instance, algorithms for a specific algorithm type.

The key responsibility of the populate node algorithm is to receive the tree code, node name and the parent context value (or values), determine if there is data that should be shown for this node, and return the a list of the context fields and values applicable to each row in the node. The following highlights additional optional information that may be returned by the algorithm:

- Entity information. If the node is related to a specific entity, the maintenance object code and primary key values may be returned. These are used for determining default information for the node, including navigation details.
- A display icon value. This is used to override the icon defined on the tree node and/or the default icon for the entity as defined on the maintenance object or business object.
- An information string. This may be used to override the default information string for an entity or to display stand-alone text such as a summary or description of the child nodes to follow.
- Override broadcast details if needed for a specific node. For example, set specific broadcast field names and values when they differ from the related entity's primary keys or indicate that broadcasting is not applicable for a node instance even though broadcasting is enabled by the node's definition.

Copyright © 2024, Oracle and/or its affiliates.

Note: The populate node algorithm is responsible for returning all the context values required to populate its child nodes. For an informational or summary node, this may require including the node's parent context fields in its own context. For example, in the script tree, the summary node describing the number of steps is followed by a list of steps for the script. The summary node populate node algorithm needs to include the script code passed in as parent context in its output context list, in order for the steps to be correctly retrieved.

Information Override Algorithm

The information override algorithm has two main purposes:

- Providing alternative information text for a tree node when the information provided isn't suitable for your implementation.
- Retrieving contextual insights related to the tree node entity, to be included in the node's information text. Only badge insight types are currently supported.
- Override broadcast details if needed for a specific node. For example, set specific broadcast field names and values when they differ from the related entity's primary keys or indicate that broadcasting is not applicable for a node instance even though broadcasting is enabled by the node's definition.

The override information algorithm receives the tree code, node name, maintenance object and prime key values for the node. The following highlights the information that may be returned by the algorithm:

- A display icon value.
- An information string. This will be used to override the default information string in the node's formatted string.
- Full node text. This will be used as the full text for the display, overriding the formatting defined on the node.
- A list of badge insight types and their context.

Note: If the information override algorithm returns insight types for the entity, it is also responsible for returning all the context values required for those insight types. Refer to Contextual Insights for more information.

Preparing a Node for Display

The following section highlights key steps in fetching a node's data and formatting it for display.

Determining Node Content

The following steps are performed to retrieve the components making up the tree node display:

- The Populate Node algorithm is called to determine all instance nodes for the tree node along with their related details.
- The Information Override algorithm is called for each node instance to allow for optional override of presentation related details.

The node display contents are determined as follows:

- The icon is determined from the following sources, in order: tree node definition, override information algorithm value, populate node algorithm value, maintenance object / business object default value.
- Broadcast usage is determined by the setting on the tree node definition.
  - This setting can be overriden by either the Populate Node algorithm or the Override Information algoirthm (which takes precedence) to indicate that broadcasting is not applicable for a specific node instance.
  - By default the broadcast fields for a node are the primary key names of its related entity. Broadcast field names can be overriden by either the Populate Node algorithm or an Override Information algorithm where the latter takes precedence.
- The information string is determined from the following sources, in order: override information algorithm value, populate node value, maintenance object / business object information algorithm. If the string is derived from the MO/BO algorithm, the short information text is used when available else the standard information is used.
- Context menu usage is determined by the settings on the tree node definition.
- The system supports a different style for header rows, including a larger font. Header style usage is determined by the setting on the tree definition and is only applicable if the node is a root node with only one instance.

Copyright © 2024, Oracle and/or its affiliates.

Node Text Format

The text for nodes referencing a system entity commonly includes some combination of a label, the entity key and an information string. The node configuration provides a label field and a substitution format string field which may be used to define the node text format. The format string defines which of the components should be included in the text, the order of those components and any surrounding text and HTML markups. For example, a format string of '**%label - %key - %info**' will result in a node text like the example below:

If the format string is blank, the default text will include '**%info**' only. If an override information algorithm is configured that returns the full node text, the text is presented without further formatting.

Auto Expansion

If the tree node definition indicates that the node should be expanded, the system will automatically fetch and display the data for its child node. This processing continues until there are no further child nodes marked for expansion.

Base Delivered Trees

Trees are system data and your product may already deliver several trees out of the box. You may choose to customize a base delivered tree in the following ways:

- You may add your own nodes to an existing tree definition.
- You may override the information string for a node by supplying your own **Information Override** algorithm. If a base override algorithm is configured on the node, your custom algorithm should be plugged in at a higher sequence number, to take precedence over the base algorithm.

Rendering a Tree

To render a tree on the UI, you would include the ou-tree web component in a UI map that references the tree definition.

Note: Refer to Include a Web Component for more information.

Defining Trees

This portal is used to view and maintain information about a tree definition. Trees are commonly used to display hierarchical relationships between objects in the system. Refer to Trees for background information about defining trees.

You can access the portal from **Admin** > **System** > **Tree**. You are brought to a query portal with options for searching for a specific tree definition. Once a tree has been selected you are brought to the maintenance portal to view and maintain the selected record.

The following zones will appear as part of the portal's **Main** tab page

- **Tree.** This zone provides information about the selected tree.
- **Tree Nodes.** This zone lists the nodes for this tree in hierarchical order. Use the zone actions to add or maintain a node in the tree.
- **Tree Node.** This zone displays the details for a tree node. It is only visible if a tree node name has been broadcast from the Tree Nodes zone.

Contextual Insights

Contextual Insights provide a powerful way to surface focused intelligence and alerts to users in the context of the core screens for any applications built on Oracle Utilities Application Framework. There are multiple options for the visual patterns when surfacing Contextual Insights and the actions that can be performed by users are easily configurable as

Copyright © 2024, Oracle and/or its affiliates.

well. The subsequent topics provide detail about the Insight Type and Insight Group objects that are used to configure the information that may then be displayed in appropriate screens.

Insight Types

The insight type object contains all the detailed configuration for displaying one or more alerts, or important information for a record in context. The following sections provide more information about all the configuration managed by the insight type.

Insight Algorithm

Each insight type is used to display some information, typically related to an object in context. It is the responsibility of the insight type algorithm to receive the context value (or values), determine if there is information that should be shown for this context, return the text to display.

The following highlights additional optional information that may be returned by the algorithm:

- A secondary message. This is only applicable to some visual structures where more than one line of information may be displayed.
- A severity value. This is used to override the visual display configuration if different emphasis should be used for different severity values. Refer to Override Configuration for more information.
- Entity information. If the insight is related to a specific entity, the maintenance object code and primary key values may be returned. This is useful if the insight type configures a BPA or user action algorithms that may perform logic for this entity.
- Navigation and User Action information. This is only applicable to visual structures that support user actions. The algorithm must return the list of all the actions that should be made visible on the insight. If the action includes navigation, the algorithm should return the navigation option and navigation context key and value pairs. Additionally, there is configuration for each action that must be defined. Refer to User Actions for more information.

Each algorithm can return multiple insights, if applicable. For example, if the insight is about To Do entries related to the object being displayed, you could design your insight algorithm to return a single insight summarizing all the To Do related information. Or you could choose to return separate insights based on To Do information so that you provide more granular information and customize actions or navigation based on the different conditions. For example, if there are To Do Entries that the current user may perform actions on and others that current user cannot update, the algorithm could return two separate insights: one with actions and one without.

Note: Preview mode. The insight type maintenance page includes an Insight Preview section that shows the output from the insight algorithm when invoked with an insight mode of "Preview". Every algorithm should include logic that returns appropriate text when called in this mode so that something is displayed in that section.

Visual Structures

Each insight type defines a visual structure which dictates what kind of information may be shown. The system has some fixed layout expectations for each visual structure type. The insight type configuration is used to provide the source for the possible layout.

Note: All the insights use color options as part of their configuration. Refer to Color Option for more information.

Badge Insight

The simplest visual structure is the badge. It is used to display a single line of quick text with a solid background (for visibility) that is meant to alert the user to important summary information or attributes.

Inline Insight

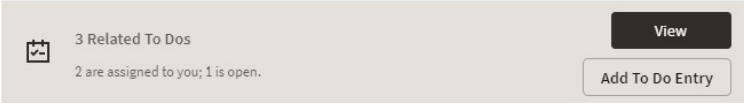
Copyright © 2024, Oracle and/or its affiliates.

The inline insight defines an icon and a single message. This is meant to be more descriptive than a badge. For this type of insight, a default light background is used so the recommendation is to select a color option for both the text and the icon that has only a foreground color defined or has a light background color.

List Insight

The list insight are intended to exist in a list of insights for the object. It defines an icon and can support two lines of text. In addition, this type of insight supports buttons or user actions. Refer to User Actions for more information. Note that for a list insight, if there is one valid action or 2 valid actions, the action buttons are shown. If there are three or more valid actions, one action is shown and then an 'Other Actions' button is visible. Clicking this shows the other available actions.

Note that the background color for the insight is used as the color for the full insight snippet. The text and the icon are shown on this background. The icon's background color is ignored.

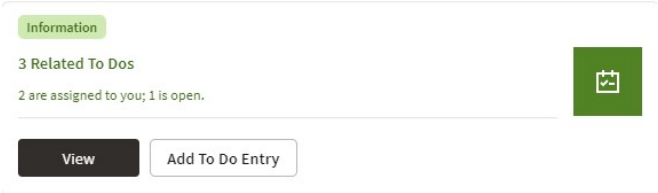


Card

The card insight is the most information rich, with up to two lines of text, user actions, a large icon to call attention and header text. The header text is chosen from a dropdown list of valid options. The product provides out of the box header text of "Information", "Warning" and "Error". Implementations may add additional values to this using the lookup field **HEADER_CAT_FLG**.

Please note the following about the treatment of the various components of the card insight:

- The header text is rendered like a badge insight.
- For the message text displayed in the card, the background color is ignored so a color option with no background color or a light background color should be chosen.
- For the icon, the background color is used to create a square area for display and the icon is rendered in the foreground color.



Override Configuration

Each insight supports default visual setting for the various components of the insight. This includes the color option for the applicable components along with icons and header text, where applicable. In each case, override visual settings may be defined for a given severity number.

For example imagine the algorithm for a badge insight that returns a message with the count of related To Do entries uses a different color option if there are more than 10 related records. To do this, the algorithm can return a severity number and then the configuration user can define the override color option for this severity number.

User Actions

The list and card insights support the definition of user actions that may be presented as buttons when displaying the insight. The actions may do one or more of the following:

Copyright © 2024, Oracle and/or its affiliates.

- Perform algorithm logic. One or more user action algorithms may be associated with an action button. This should be used to perform server logic that does not require any user interaction. For example, you can provide an action to update the status of a record related to the insight. These algorithms are configured on the Insight Type.
- Navigate to another page. For this type of action, the insight algorithm must provide the navigation context and the navigation option to launch when clicking the button. It's possible to configure navigation and configure user action algorithms. In that case the algorithms are run first and then the navigation occurs.
- Launch a BPA. This is used if there is user interaction needed to perform additional logic and / or if there is conditional navigation such that the logic is needed before navigating. It's possible to configure a BPA and configure user action algorithms. In that case the algorithms are run first and then the BPA is launched. If the insight algorithm also provides navigation for this action, it is ignored in favor of the BPA. The BPA is configured on the Insight Type.

The list of valid actions to display at run time must be returned by the insight algorithm. This allows for the algorithm to have control over which actions to display, in case there are some actions that are conditional. In addition, each possible button requires configuration related to what text should display and the style of the button.

Button Text

The button text to display is chosen from a button text lookup. The lookup name is **INSIGHT_ACT_FLD_FLG**. Implementations may add additional values to this lookup to include new button text. In addition, if you would like to change button text on any product delivered button, you can override the lookup value description.

Action Class

Each action button defines an action class. Any action configured with the action class value of **Positive** value is shown first and is styled differently for emphasis. Other action class values are for information purposes.

Base Delivered Insight Types

Insight types are system data and your product may already deliver several insight types out of the box. You may choose to include one or more insight types in an Insight Group.

Rendering an Insight

To render an insight for an insight type, you would include the ou-insights web component in a UI map that references this insight type. This is how the Insight Preview section of the insight type maintenance page renders the preview.

Note: Refer to Include a Web Component for more information.

Defining Insight Types

The insight type entity is used to define the configuration of a specific type of alert to display extra information related to one or more records in context. Refer to Insight Types for a detailed description of the supported functionality.

To view or maintain insight types, open **Admin** > **General** > **Insight Type.**

This is a standard All-in-One portal.

The information captured on the insight type depends on the record's business object. Refer to the embedded help text for more information.

Related Insight Groups

If there are any insight groups that reference this insight type, the **Related Insight Groups** zone displays the linked insight groups.

Insight Groups

An insight group may be used to include several insight types together onto a single map. When defining an insight group, you configure the visual structures that the insight group should include. Only insight types for the configured visual structure are available.

Copyright © 2024, Oracle and/or its affiliates.

Insight Class

Each insight group must reference a unique insight class. The insight class is defined using a lookup. The lookup field is **INSIGHT_CLASS_FLG**. Your product may already deliver base values for insight class. To display insights in a user interface your UI map would include the ou-insights web component. This component includes syntax to reference an insight class and the logic will find the insight group for that class and render all the insight types linked to that group.

Because the insight class is used as the reference in the web component rather than the insight group, it allows your product to supply base logic to show insight group information in a base delivered user interface. The base logic defines an insight class in base and references it on a base UI map. Your implementation can then define the insight group for this insight class and choose the insight types that should be rendered in that map. Refer to your specific product documentation to find out if there are any base delivered insight classes.

If your implementation identifies a new use case where one ore more insight types should be displayed, once the insight types are configured, do the following:

- Define a new Insight Class lookup value.
- Define an Insight Group that references this insight class and configure the appropriate insight types.
- Include the appropriate web component in the map for your user interface where you wish to display this information.

Defining Insight Groups

The insight group entity allows implementations to define multiple related insight types that should be displayed together in a given user interface map or zone. Refer to Insight Groups for an overview of insight group configuration and functionality.

To view or maintain insight groups, open **Admin** > **General** > **Insight Group.**

This is a standard All-in-One portal.

The information captured on the insight group depends on the record's business object. Refer to the embedded help text for more information.

Related Insight Types

The **Related Insight Types** zone displays all the insight types that are currently linked to the group. The Edit link in the zone header allows you to add or remove insight types. Only insight types that reference one of the **Valid Visual Structures** may be selected.

Color Configuration

The topics in this section describe some options for configuring colors to be used in displaying information in the user interface.

Defining Colors

The system provides the extendable lookup Color (**F1-Color**) to define valid colors to be used in the system. The base colors provided are ones related to the standard color palette used in Oracle products.

At this time, zones and UI maps that define specific colors do not support any syntax to reference a color extendable lookup. In the meantime, colors applied at the zone or map level can still reference HTML colors or reference hex codes directly.

Color Options

The color option entity is used to define a combination of colors to be used in user interface components. The system provides a standard set of color options used for contextual insights.

Copyright © 2024, Oracle and/or its affiliates.

To view or maintain color options, open **Admin** > **System** > **Color Option.**

This is a standard All-in-One portal.

The information captured on the color option depends on the record's business object. Refer to the embedded help text for more information.

Defining Display Icons

Icons are used to assist users in identifying different types of objects or instructions. A limited number of control tables allow administrative users to select an icon when they are configuring the system. Select **Admin** > **System** > **Display Icon** to maintain the population of icons available for selection. You are brought to a query portal with options for searching for a specific display icon. Once a display icon has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The latest standard is to use Scalable Vector Graphic (SVG) icons. For these types of icons, the Icon Type is **Variable (SVG)**. The product also supports icons in the .gif or .png format. For these types of icons, the Icon Type defines a pixel size.

URL describes where the icon is located (folder) and the icon name (file name).

Note that there is a standard folder structure for delivered assets such as images. The structure is **{DefaultWebApp}/{owner}/assets/images|svgs/**

- {Owner} - this is the owner of the icon. The **/cm** folder is used for custom icons.
- svgs - all images that are of type svg are in this folder.
- images - all files that have a file suffix of jpg, gif, png or other types of image formats are in this folder.

When using the above folder structure, then the URL only needs to contain the icon name (file name). The correct full URL is determined automatically at runtime. Images can optionally be placed in other folders under the DefaultWebApp. However, this is not recommended but if used, the URL must contain the full path to the icon (e.g. /cm/assets/newImages/myIcon.gif)

Custom Look and Feel Options

The default look and feel of the application can be customized via feature configuration and cascading style sheets. The base product is provided with a **Custom Look And Feel** Feature Configuration type. You may want to set up a feature configuration of this type to define style sheet and UI Map help options.

User Interface

The base product allows for the conditional inclusion of custom style sheets into the system style set. Custom styles may override any style provided by the base product. The style sheet may also include new styles for use in customer zone definitions. Use the **Style Sheet** option on the **Custom Look And Feel** Feature Configuration to define your custom style sheet.

Note: Some styles cannot change if they are part of the HTML code.

CAUTION: Implementers must ensure that the customized user interface is stable and scalable. Changing font, alignment padding, border size, and other user interface parameters may cause presentation problems, like scrollbars appearing or disappearing, cursors not working as expected, and unanticipated look and feel alterations of some layouts.

Copyright © 2024, Oracle and/or its affiliates.

UI Map Help

A tool tip can be used to display additional help information to the user. This applies to section elements as well as individual elements on a map zone or UI Map. Refer to the tips context sensitive zone associated with the UI Map page for more information. The **Custom Look And Feel** Feature Configuration provides options to control the following:

- Whether UI Map Help functionality is turned on or off. By default it is turned on.
- Override the default help image with a custom image
- The location of the help image, either before or after the element.

Fastpath: Refer to the feature configuration for a detailed description of each option.

Chapter 4

Database Tools

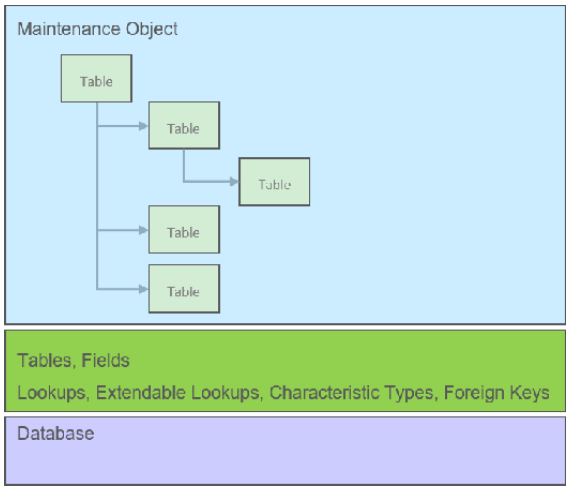
Topics:

- Understanding The Data Model
- Defining Tables
- Defining Fields
- Defining Maintenance Objects
- Defining Lookups
- Defining Extendable Lookups
- Defining Characteristic Types
- Defining Foreign Key References
- Audit Trails
- Bundling
- Revision Control
- Information Lifecycle Management

This section describes a variety of database tools that are supplied with the your product.

Understanding The Data Model

The application uses meta-data to describe its data model, from plain description of its tables and fields to more complex hierarchies of maintenance objects.



The following points briefly outline key aspects of application's data model as illustrated in this diagram:

- Table and Field records describe all the tables maintained by the application.
- Valid values for certain fields are defined using corresponding Lookup and Extendable Lookup records.
- Characteristic types may be defined to support an extension of a maintenance object via its characteristics table.
- Maintenance Objects define the physical structure of an entity. Business objects further refine maintenance objects to describe business entities. Refer to The Big Picture of Business Objects for more information.
- Foreign Key References define ways for searching and navigating to the maintenance portal of an entity.

The following sections further discusses concepts and guidelines related to the data model.

Fields

A Field definition describes the metadata needed by the application to support the capture and display of a piece of information. Fields may correspond to physical columns on database tables or represent any type of element used for calculation or presentation purposes.

The definition includes the following key details:

- Whether the field refers to a physical column on a database table or used as a work field.
- The type of data the field holds.
- The label to use when the field is presented.
- The help text to display when the user clicks on the help icon seeking additional information about this field.
- and more

The following sections further discusses concepts and guidelines related to defining fields.

Physical vs. Work Fields

A Field definition record exists for every physical column on a database table that is used by the application. Refer to Understanding Tables for more information.

A Field that does not represent a physical column on a database table is considered as work field and needs to be marked as such. Work fields are typically used as labels as well as describe internal elements used in business rules.

Copyright © 2024, Oracle and/or its affiliates.

Data Type Information

The type of data captured by a field is defined using the following attributes:

- **Data Type** indicates the basic type of value the field holds. Valid values are **Character, Character Large Object, Date, DateTime, Number, Time, Varchar2** and **XML Type**.
- **Extended Data Type** is used to further define the type of data for certain data types. Valid values are **Currency Source, Day of Month, Duration, Money, Month of Year, Flag, Switch** and **URI**.
- **Precision** defines the length of the field. In the case of variable length fields, it is the maximum length possible. For number fields that include decimal values, the precision includes the decimal values.
- **Scale** is only applicable for number fields. It indicates the number of decimal places supported by the field.
- **Sign** is only applicable for numbers. It indicates if the data may contain positive or negative numbers.

Data Type Inheritance

A Field can be defined to inherit its data type attributes from the definition of another field. The field to inherit the attributes from is referenced as the **Base Field**. When using a base field reference, the data type attributes are defined once, on the base field, and all the fields inheriting their data type information from it are kept consistent with that definition as it may change over time. Any change of a field's data type attributes is automatically reflected on all the fields that reference it as their base field.

Data type information may either be specified directly on the field itself or by referencing a base field. The system does not support a way to partially override the base field's data type information.

It is recommended to leverage data type inheritance whenever possible, especially when the reason for creating a new field is to provide an alternate label for the original field in a certain context. In that situation the new field should reference the original field as its base field and provide the new labels and help text as needed.

Labels and Help Text

For fields that appear on a user interface, the description acts as a label to be used for presentation purposes. As per the product's multi-language support, a field is associated with a label for each of the languages enabled in the system.

A field's help text may be used to provide the user with additional information on how the field is used. When populated for a field that is visible on a user interface that supports display of embedded help, a help icon appears for the field, presenting the help text to the user when clicked.

Overriding Labels and Help Text

The label and help text provided for a product owned field may be overridden by your organization using the corresponding override fields.

The field's label and help text can be overridden for a specific table by populating the corresponding override information as part of the table / field definition. However, this override is not used in portal based user interfaces. It is only applicable if the field is displayed on fixed page user interfaces. For fixed pages, if the override label at the table / field level takes precedence over the override at the field level.

Alternate Labels

There are contexts where a separate label may be needed for the "date" and "time" parts of a date/time field. Such an example is when the field is used by the operational analytics product. For example, for a "Completion Date/Time" field you may provide the context label "Creation Date" for the **Date** context and "Creation Time" for the **Time** context. These context labels are optional.

In the same way, date fields, time fields and date/time fields may be associated with a **Date and Time Base** context label, which is a label without a reference to the word "Date", "Time" or "Date / Time". For example, for a "Completion Date/Time" field, the date and time base label may be "Completion". This label that can be used in combination with date or time attributes such as day, month, year, hours, minutes etc.

Overrides labels are available for these additional context labels.

Copyright © 2024, Oracle and/or its affiliates.

Tables

A Table definition describes the metadata needed by the application to work with and access a database table or view.

The definition includes the following key details:

- The table's description in the application.
- The list of fields defined on the table and any foreign key constraints they establish with other tables.
- The table's prime key and how it is generated.
- Whether the table holds administrative, master or transactional data.
- Whether the table is released with product owned system data rows or not.
- The table's role in the conversion process.
- How should table content be upgraded.
- and more

The following sections further discusses concepts and guidelines related to defining tables.

System Tables

Tables released with rows that are owned by the base product are referred to as system tables. These tables include the **Owner** (OWNER_FLG) field to capture the owner of each row.

Refer to System Data for more information about naming conventions and standards around row record ownership.

The **Upgrade** setting on the table definition record controls what happens to the rows in a table when the system is upgraded to a new release. The most common setting allows customer owned rows to be merged with rows owned by the base product. Other settings may keep the content of the table as is, i.e. rows are not touched by the upgrade, or fully refresh the content of the table with base product rows.

Some fields on certain system owned tables are explicitly marked to allow customization. This means implementation are allowed to change these specific fields for a base owned record. Changes to the field value of one of these types of fields by an implementation are maintained when upgrading to a new version of the product.

Designated Table Roles

Some tables are designed to support the following application roles:

- **Key Tables**
  - A key table holds a copy of a table's prime key only and is only applicable to tables with system generated keys.
  - Tables with a system generated keys reference their Key Table as part of their Table definition record.
  - Records in the key table remain in the system even when the related entity is archived, preventing the same key from being allocated to a new record. Retaining the key allows the system to keep the integrity of any references that may still exist to the archived entity.
- **Language Tables**
  - To support multi-language implementations, all labels and descriptions that are presented to a user are kept in a designated language table where a row exists for each enabled language.
  - The product is released with the English content provided in the various language tables. An additional row is added to these tables for each enabled language along with the translated content for that language. Refer to Support For Different Languages for more information.
  - Tables associated with language sensitive fields reference their corresponding Language Table as part of their Table definition record.
- **Characteristics Tables**
  - Many entities in the system support one or more Characteristics tables, which are used to capture additional fields for the object that are not already supported by the entity's provided attributes.
  - A Characteristics table is designed to capture characteristics applicable to its specific entity only. The entity associated with a Characteristics table is referenced on the Table's definition record.

Copyright © 2024, Oracle and/or its affiliates.

  - Refer to Setting Up Characteristic Types & Values for more information.
- **Entity Log Tables**
  - Many entities in the system support the ability to capture lifecycle and other important business events on a designated log table.
  - A log table captures the event details in the form of an application message and stores any parameters for the message in a child log parameters table. All log and log parameters tables follow a predefined structure allowing them to be consistently and efficiently implemented.
  - Typically log tables do not participate in the conversion process.

Table Constraints

Foreign key relationships between tables are not enforced at the database level but rather at the application level, providing the application more flexibility around data manipulation and referential integrity.

The following types of constraints may be defined on the Table's definition record:

- **Primary Key** represents the field or set of fields that represent the unique identifier of a record stored in a table.
- **Logical Key** represents an alternate unique identifier of a record based on a different set of fields than the Primary key.
- **Foreign Key** represents a field or set of fields that specifies identifying and non-identifying relationships to other tables in the application. A foreign key constraint references the primary key constraint of another table.
- **Conditional Foreign Key** represents rare relationships between tables where a single field (or set of fields) may reference multiple primary key constraints of other tables within the application as a foreign key.

The system leverages a table's foreign key constraints information to performs referential integrity validation when rows in this table are deleted. These validations can be disabled by checking off the **Enable Referential Integrity** field on the Table definition record.

Table Indexes

The indexes delivered by the base product for a given table are visible in the index metadata. In addition, table index information may be defined as part of the cloud services support for generation of custom tables in certain use cases. Refer to the section Generating Custom Tables below for more information.

Primary Key Type

A table’s primary key may be of the following types:

- **User-defined** means the user specifies the key when a record is added. This type of key is typically used for configuration and administrative tables.
- **System-generated** means a program generates a random numeric key for the record when it is added.
  - This type of table is typically associated with a key table. Refer to the **Designated Table Roles** section for more information.
  - System generated keys may be fully random or partially inherited from a parent table’s key for performance reasons. Refer to the **System Generated Keys** for more information.
- **Sequential** means a sequence number is incremented whenever a record is added to the table. The next number in the sequence determines the key value.
- **Database Native Sequential Key** means a unique number is assigned using a database sequence object.
- **Other** means some other business rule allocates the table's prime-key or the prime key is made of multiple parts.

System Generated Keys

In general, randomly generated numeric system keys are used to attempt to evenly distribute records across a full range of possible IDs. Batch programs that use multiple threads will typically divide the threads using ID ranges and evenly distributed keys will help spread out the work across these threads.

Copyright © 2024, Oracle and/or its affiliates.

If a table using a system generated key is a child of another table, the child table's key may **inherit** a portion of the random number from its parent table's key. Key inheritance for child records further extends the performance benefit. When considering partitioning, the recommendation for DBAs is to range partition data based on the primary key so that different batch threads operate on different partitions which reduces contention for hot blocks. Ideally the number of batch threads will be an exact multiple of the number of partitions. Batch programs that insert child data (for example batch Billing creation) also benefit from this design especially when the child tables are partitioned in the same way. The parent is often the driver of the batch process. If this is multi-threaded, then each thread is processing a set of parent records in a given ID range and all child records are being inserted into the same ID range.

While most entities with system generated keys either use fully random keys or partially inherited keys from a single parent, entities may be designed to implement more specific key generation rules.

When key inheritance is used, the table definition of the child table defines the following information:

- The number of most significant digits from the parent record's primary key to use as the **prefix** for the child record's key value.
- The **parent key field name** to inherit the key from. This information is provided for display and documentation purposes only and is only provided when the inheritance rule implemented by the entity may be documented this way. Typically the child key value is inherited from a single parent key but in some cases the inheritance rule may be more complex. For example, different types of records in the same child table may inherit their key from different parent key fields. In this case, multiple parent field names may be listed to describe the various inheritance options. When the rule is too complex, this information may not be provided on the table definition record.

Conversion Options

The conversion process manages the process by which master and transaction legacy data, loaded into a staging schema, is validated and assigned new system generated keys before it is copied to the production schema. Refer to Understanding The Conversion Process for more information.

The tool classifies tables into the following roles:

- Administrative tables. These only exist in the production schema. Instead of a table, a view to the production table exists in the staging schema. These views are necessary to enforce foreign key references when data is validated in the staging schema
- Convertible tables that retain their prime keys. Data conversion is supported for these tables and their prime key values remain unchanged.
- Convertible tables that require a new system generated key to be assigned. Data conversion is supported for these tables and their prime key is reassigned by the system during conversion.
- Not convertible tables. Assigning new system generated keys for a record and replacing all foreign key references it might have with the new key is a complicated process that may not be supported for all tables.

. The **Data Conversion Role** setting on the table definition record controls if / how the table is used by conversion processing.

Date/Time Configuration

Fields capturing date/time information may be designed to store data in either local legal time or local standard time. Local legal time is captured as entered and displayed as captured in the database. It also assumes the date/time information refers to the time zone defined on the installation options. Refer to Standard vs. Legal for more information.

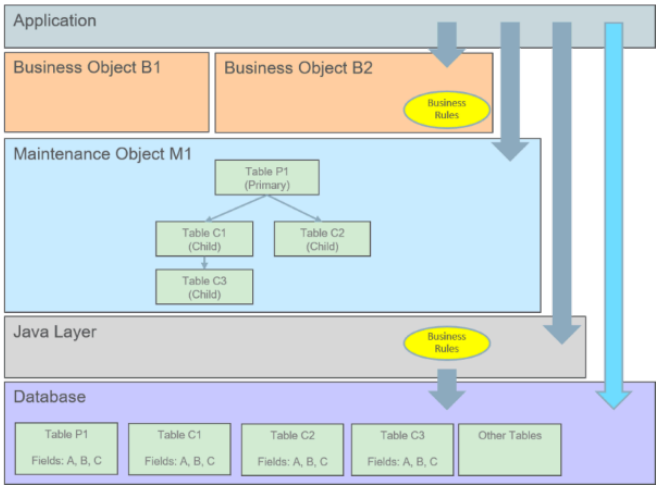
Generating Custom Tables

For cloud customers and for certain use cases, the product supports defining table metadata for custom tables and generating the related DDLs for that metadata in the database. For these custom tables, there is a Generation Status visible on the Table metadata that identifies where the Table is in the lifecycle for this supported feature. In addition, the index tab on the table portal is used to define custom indexes that are included in the DDL as part of this feature. If you are a cloud customer and would like more information, refer to the Cloud Services Implementation Guide.

Copyright © 2024, Oracle and/or its affiliates.

Maintenance Objects

A maintenance object defines the configuration of a given “entity” in the system. It includes a primary table and any additional child tables that together capture the physical data for the entity. The application reads, adds, updates and deletes an entity as a single object by invoking its maintenance object service.



The following points briefly outline key aspects of application’s interaction with maintenance objects as illustrated in this diagram:

- Database tables are maintained solely via the application’s Java layer. Direct updates to a table via its Java class is not common and typically reserved to special processes.
- A maintenance object orchestrates the update to an entity’s primary table and its child tables as a single object.
- A business object defines a logical representation of an entity, including all or a subset of the maintenance object’s tables and fields.
- The more common way to maintain an entity is via its maintenance object, either directly or via one of its business objects. Most entities support the use of business objects and as such their maintenance is typically made via their business objects.
- Direct access to database tables is supported for query purposes only.

The following sections further discusses concepts and guidelines related to defining tables.

Maintaining An Entire Object

A maintenance object includes a single primary table and one or more child tables where each child table can further have nested child tables. The table structure of a maintenance object represents the scope of data that should be maintained together as a single object:

- When an object is added, data in added to each corresponding table.
- When an object is updated, data in updated in each corresponding table.
- When an object is read, data in retrieved from all the tables.
- When an object is deleted, data in deleted from all the tables.

Algorithms

Several algorithm plug-in spots are also defined on the maintenance object, allowing for business rules that govern all records for this maintenance object.

Options

The maintenance object includes options that define important information related to the maintenance object that may be accessed for logic throughout the system.

Copyright © 2024, Oracle and/or its affiliates.

Maintenance Objects vs. Business Objects

A maintenance object defines the physical structure of an entity as stored in the database. Different business objects may be defined on top of a maintenance object to represent different types of such entity. A business object includes only the relevant information applicable to the specific type of entity as well as enforces business rules applicable only to that type.

When a maintenance object supports business object configuration, the system uses the **Determine Business Object** maintenance object algorithm to determine the identifying business object that controls a given entity's data structure and rules.

Refer to The Big Picture of Business Objects for more information.

Defining Valid Values

The product provides several options for defining valid values for a column on a table:

- Lookup
- Extendable Lookup
- Control Table

The following provides more information about the functionality of each of the options available for defining valid values for a column.

Lookup

The simplest mechanism for defining valid values for a column on a table is via the Lookup table. This is sometimes referred to as a "simple" lookup to distinguish it from an extendable lookup (described below). Using the lookup table, you can define valid values and their descriptions. When choosing a valid value that is defined by a lookup, a dropdown UI metaphor is used.

The following highlights functionality related to lookups:

- Lookups are associated with a Field. The field is defined as a character data type with an extended data type of **Flag**. The field's label serves as the description for the prompt to select the valid value.
- The lookup code is limited to four characters and must be all uppercase. If there is any functionality where a valid value in the application must match valid values in an external system, the lookup table may not be the appropriate choice.
- The lookup table does not support additional attributes to be defined for each value. This option is only appropriate when a simple code and description pair is needed.
- The product may also use Lookups to define valid values for functionality unrelated to a column on a table. For example, an algorithm plug-in spot may define an input parameter that supports one or more valid values. The plug-in spot may define the valid values using a lookup, allowing for a simple way to validate the value supplied when invoking the algorithm and to document the valid values.

Fastpath: For more information, refer to Defining Lookup Options.

Extendable Lookup

The extendable lookup provides a way of defining valid values for a column with additional capabilities that are not supported using the Lookup table. When choosing a valid value that is defined by an extendable lookup, a dropdown UI metaphor is used.

The following highlights functionality related to extendable lookups:

- Each Extendable Lookups is defined using a business object.
- A field should be defined for the extendable lookup code. The field defines the label for the lookup code and defines the size of the lookup code. The size is determined based on the business use case. In addition, there are standard

Copyright © 2024, Oracle and/or its affiliates.

fields included in all extendable lookups, including a description, detailed description and an override description (so that implementations can override the description of base delivered values).
- The extendable lookup may define additional information for each value if warranted by the business requirement. See Additional Attributes for technical information about additional attributes.

Fastpath: For more information, refer to Defining Extendable Lookups.

Control Table

There may be scenarios where a list of valid values warrants a standalone maintenance object, which is considered an administrative or control table object. When choosing a valid value that is defined by a control, either a dropdown UI metaphor or a search metaphor is used, depending on how it has been designed.

The following points highlight some reasons why this option may be chosen:

- The records require a lifecycle such that BO status is warranted.
- The additional attributes are sophisticated enough that they warrant their own column definition rather than relying on using CLOB or flattened characteristic. For example, if a list of information needs to be captured with several attributes in the list and the information in the list needs to be searchable.

In this situation, if a product has provided a control table for this type of functionality, it will be documented fully in the appropriate functional area. If an implementation determines that a custom control table is warranted, all the standard functionality for a maintenance object is required: database tables, maintenance object metadata, appropriate Java maintenance classes, portals, zones, etc. Refer to the Software Development Kit for more information. No further information is provided in this section for this option.

Characteristic Types

Many maintenance objects in the system support characteristics tables, which are used to capture additional fields for the object that are not already supported by the object's provided attributes. Each characteristic is associated with a characteristic type, which defines attributes of the field you wish to capture.

While each characteristic is stored in a separate row in the characteristic table, the user interface metaphor used to capture and display them differ based on the type of page that is used to maintain the object.

- For portal based pages, the business object drives the display and maintenance of the object. The display and maintenance of each characteristic is determined by the business object's user interface design. For these types of pages, it is recommended that characteristics are defined as flattened elements allowing the user interface to display the characteristic as if it is another field. Refer to The Flattening Nodes and Attributes for more information about flattening characteristics on a business object schema.
- There are some fixed pages in the system that do not support customization of the user interface. For these objects, the characteristics are displayed / maintained as a generic list.

The topics in this section describe various aspects of a characteristic type definition and usage.

Characteristic Value Types

Every characteristic referenced on an object references a characteristic type. The characteristic type controls the validity of the information entered by a user when they enter the characteristic's values. For example, if you have a characteristic type on user called "skill", the information you setup on this characteristic type controls the valid values that may be specified by a user when defining another user's skill.

When you setup a characteristic type, you must classify it as one of the following categories:

- **Predefined Value.** When you setup a characteristic of this type, you define the individual valid values that may be entered by a user. A good example of such a characteristic type would be one on User to define one or more predefined skills for that user. The valid values for this characteristic type would be defined in a discreet list.
- **Ad hoc Value.** Characteristics of this type do not have their valid values defined in a discreet list because the possible values are infinite. Good examples of such a characteristic type would be ones used to define a user's birth date or

Copyright © 2024, Oracle and/or its affiliates.

their mother's maiden name. Optionally, you can plug-in an algorithm on such a characteristic type to validate the value entered by the user. For example, you can plug-in an algorithm on a characteristic type to ensure the value entered is a date.

- **Foreign Key Value.** Characteristics of this type have their valid values defined in another table. For example perhaps you want to link a user to a table where User is not already a foreign key. Valid values for this type of characteristic would be defined on the user table. Before you can create a characteristic of this type, information about the table that contains the valid values must be defined on the foreign key reference table.
- **File Location.** Characteristics of this type contain a URL. The URL can point to a file or any web site. Characteristics of this type might be useful to hold references to documentation / images associated with a given entity. For example, the image of a letter sent to you by one of your customers could be referenced as a file location characteristic on a customer contact entry. When such a characteristic is defined on an entity, a button can be used to open the URL in a separate browser window.

  File location characteristic values must be entered in a "non-relative" format. For example, if you want to define a characteristic value of *www.msn.com*, enter the characteristic value as `http://www.msn.com`. If you omit the `http://` prefix, the system will suffix the characteristic value to the current URL in your browser and attempt to navigate to this location when the launch button is pressed. This may or may not be the desired result.

  > **Note:**
  >
  > Due to browser security restrictions, opening URLs using the file protocol ("file://") from pages retrieved using http does not work. If the file protocol is used, the browser either does not return properly or an error is thrown (e.g., "Access Denied", which usually results from cross site scripting features added for security reasons). This issue has no known workaround. To comply with browser security standards, the recommendation is to move the target files to an FTP or HTTP server location to avoid protocols that are subject to browser security restrictions.
  >
  > Also note that the functionality described in the topics for Referencing URIs do not apply to this value given that the browser is responsible for connecting to the URI and does not go via server logic.

  For references to a file, the recommendation is to use the Attachment functionality to link a file to an object rather than a characteristic type of File Location. Refer to Attachment Overview for more information. The documentation related to file location remains for upgrade purposes.

Searching By Characteristic Values

For certain entities in the system that have characteristics, you may search for a record linked to a given characteristic value. The search may be done in one of the following ways:

- Some base searches provide an option to search for an object by entering Characteristic Type and Characteristic Value.
- Your implementation may define a customized search for an entity by a characteristic value for a specific characteristic type using a query data explorer.
- Your implementation may require a business service to find a record via a given characteristic value. For example, maybe an upload of user information attempts to find the user via an Employee ID, defined as a characteristic.

Not all entities that support characteristics support searching by characteristics. To confirm if a given characteristic table supports searching, verify that it includes the field **SRCH_CHAR_VAL**.

> **CAUTION:** For ad-hoc characteristics, only the first 50 bytes are searchable. For foreign key characteristics, the search value is populated by concatenating the values of each foreign key column to a maximum of 50 bytes.

For the base searches that provide a generic option to search by characteristic type and value, you can restrict the characteristic types that can be used to search for an entity. For example, imagine you use a characteristic to define a "jurisdiction" associated with a To Do for reporting purposes. If your company operates within a very small number of jurisdictions, you wouldn't want to allow searching for a To Do by jurisdiction, as a large number of To Do entries would be returned.

A flag on the characteristic type allows an administrator to indicate if searching by this characteristic type is **allowed** or **not allowed**.

Copyright © 2024, Oracle and/or its affiliates.

Data Typing an Ad-hoc Characteristic Value

An **Ad hoc Value** characteristic type may further restrict the data type of the captured value by referencing a Field. When populated, the system validates that the captured characteristic value complies with the data type definition of the referenced field.

Valid Characteristic Entities

Characteristics are stored in designated tables, each associated with a unique characteristic entity value referenced on the table definition record. A characteristic type record lists all the valid characteristic entities it is allowed to be captured on.

Customizing Base Owned Characteristic Types

Implementations may customize the following aspects of a base owned characteristic type:

- Extend the list of valid values for a base owned **Predefined Value** characteristic type to include additional custom values. Such extension is only supported if the characteristic type is defined with the **Custom** option turned on.
- Provide a custom algorithm to validate an **Ad hoc Value** characteristic type.
- Indicate whether characteristics of this type may be used in searches.
- Extend the list of entities associated with the characteristic type.

Foreign Key References

A Foreign Key Reference defines the necessary information needed to reference an entity in a certain table.

You need to set up this control table if you need to validate a foreign key value against a corresponding table. For example, if a schema element is associated with an FK Reference the system validates the element's value against the corresponding table. Refer to Configuration Tools to learn more about schema-based objects. Another example is characteristics whose valid values are defined in another table (i.e., you use "foreign key reference" characteristic types). Refer to Characteristic Types for a description of characteristics of this type.

A FK Reference is used not just for validation purposes. It also used to display the standard information description of the reference entity as well as provide navigation information to its maintenance transaction. Info descriptions appear throughout the UI, for example, whenever an account is displayed on a page, a description of the account appears. The product provides base product FK references for many of its entities as they are used for validation and display of elements in both fixed page user interfaces as well as portal based user interfaces.

An implementation may also see the need to define a foreign key reference. The following points describe what you should know before you can setup a foreign key reference for a table.

- The physical name of the table. Typically this is the primary table of a maintenance object.
- The program used by default to construct the referenced entity's info description.
- The transaction used to maintain the referenced entity. This is where the user navigates to when using the "go to" button or hyperlink associated with the entity.
- The name of the search page or zone used to look for a valid entity.

The topics in this section describe various aspects of a foreign key reference definition and usage.

Information Description Is Dynamically Derived

Typically a FK Reference is defined for a maintenance object's primary table. In this case the system dynamically derives the standard information associated with a specific referenced entity as follows:

- Attempt to determine the business object associated with the referenced entity. Refer to the Determine BO maintenance object algorithm system event for more information. If a business object has been determined, the system lets the business object's Information plug-in, if any, format the description.
- If a business object has not been determined or the business object has no such plug-in, the system lets the maintenance object's information plug-in, if any, format the description.

Copyright © 2024, Oracle and/or its affiliates.

- If the maintenance object has no such plug-in, the system uses the info program specified on the FK Reference to format the information.

Note: Technical note. The class that returns the information displayed adjacent to the referenced entity is generated specifically for use as an info routine. Please speak to your support group if you need to generate such a class.

Note: Generic routine. The system provides a generic information routine that returns the description of control table objects from its associated language table. By "control table" we mean a table with an associated language table that contains a **DESCR** field. Refer to Defining Table Options for more information on tables and fields. The java class is **com.splwg.base.domain.common.foreignKeyReference.DescriptionRetriever**.

Navigation Information Is Dynamically Derived

Typically a FK Reference is defined for a maintenance object's primary table. In this case the system dynamically derives the actual transaction to navigate to for a given referenced entity as follows:

- Attempt to determine the business object associated with the referenced entity. Refer to the Determine BO maintenance object algorithm system event for more information. If a business object has been determined, use the maintenance portal defined as its **Portal Navigation Option** business object option.
- If a business object has not been determined or the business object defines no such option, the system uses the transaction specified on the FK Reference.

Search Options

The product provides two main metaphors for implementing a user interface. For input fields that are foreign keys, search options are dependent on the metaphor used by the page in question.

- A portal based user interface is a more flexible user interface where an implementation has more options for customizing the look and feel. The base product uses UI maps or automatic UI rendering to display input fields. Elements that are foreign keys may display a search icon if the FK reference defines a Search Zone.

  Note: Defining search zones directly. It's possible for elements on a UI map to define a specific search zone directly in the HTML, rather than using the search zone defined on an FK reference. Refer to the UI map tips for more information on implementing searches using zones.

- A fixed maintenance page user interface is a page supplied by the base product where only minor enhancements, if any, can be introduced by implementations. The foreign key reference may be used in one of two ways.
  - The based product may use an FK reference to define a base element on one of these pages. If a search is available for such elements, the FK reference's Search Navigation Key is used to implement the search.
  - Entities that support characteristics typically include a generic characteristic collection UI metaphor on these types of pages. In this metaphor, a foreign key characteristic displays a search icon if the FK Reference has configured a Search Navigation Key.

Note: Not every FK reference provided with the product is configured with a search option. This may be the case if the base delivered pages use a dropdown for this foreign key rather than a search. In addition, base provided FK references that do provide a search may not be configured with both search options. It means that if linking this type of object as a characteristic, the search may not be available if the appropriate search is not configured.

Defining Tables

This portal is used to view and maintain information about an application table.

You can access the portal from the**Admin Menu** > **Database** > **Table**. You are brought to a query portal with options for searching for a specific table. Once a table has been selected you are brought to the maintenance portal to view and maintain the selected record.

Refer to Tables for more information.

Copyright © 2024, Oracle and/or its affiliates.

The following zones may appear as part of the portal's **Main** tab page

- **Table.** This zone provides information about the selected table.
- **Fields.** This zone lists all fields of the selected table and provides various actions to maintain this information.

The following zones may appear as part of the portal's **Relationship** tab page

- **Entity Relationship Diagram.** This zone provides a visual representation of the table's relationship to other tables in the application's data model.

The following zones may appear as part of the portal's **Constraints** tab page

- **Table Constraints.** This zone lists the various key constraints defined for the selected table and provides various actions to maintain this information.
- **Referred By Constraints.** This zone lists all tables referring to the selected table via foreign key constraints.

Refer to Tables for more information.

The following zones may appear as part of the portal's **Indexes** tab page

- **Table Indexes.** This zone lists the indexes defined for the selected table and provides various actions to maintain this information.

Refer to Tables for more information.

Defining Fields

This portal is used to view and maintain information about a field. For product owned fields, you may use this portal to override the label of a field on the various pages in the system.

You can access the portal from the **Admin Menu** > **Database** > **Field**. You are brought to a query portal with options for searching for a specific field. Once a field has been selected you are brought to the maintenance portal to view and maintain the selected record.

Note: When introducing a new record, carefully consider its naming convention. Refer to System Data Naming Convention for more information.

Refer to Fields for more information.

The following zones may appear as part of the portal's **Main** tab page

- **Field.** This zone provides information about the selected field.
- **Tables Using Field.** This zone lists tables referencing the field. It is not visible for work fields.
- **Lookup Values.** This zone lists and maintains valid values for the selected lookup field. The zone is only visible when the selected field is a lookup field.

The following zone may appear as part of the portal's **References** tab page

- **Field References.** This zone list various configuration entities referencing the current field.

Defining Maintenance Objects

A maintenance object defines the configuration of a given "entity" in the system. It includes the definition of the tables that together capture the physical data for the entity. In addition, the maintenance object includes options that define important information related to the maintenance object that may be accessed for logic throughout the system. Several algorithm plug-in spots are also defined on the maintenance object, allowing for business rules that govern all records for this maintenance object.

Many maintenance objects in the system support the use of business objects to further define configuration and business rules for a given record. Refer to Maintenance Objects for more information.

Copyright © 2024, Oracle and/or its affiliates.

Many maintenance objects in the system support the use of business objects to further define configuration and business rules for a given record. Refer to Maintenance Objects for more information.

Maintenance Object - Main

Navigate to**Admin** > **Database** > **Maintenance Object** to view information about a maintenance object.

Use the **Maintenance Object Query** portal to search for an existing maintenance object. Once a maintenance object is selected, you are brought to the maintenance portal to view and maintain the selected record.

Description of Page

Most maintenance objects are provided with the base package. An implementation can introduce custom maintenance objects when needed. Most fields may not be changed if owned by the base package.

Enter a unique **Maintenance Object** name and **Description**. **Owner** indicates if this business object is owned by the base package or by your implementation (**Customer Modification**).

> **Important:** If you introduce a new maintenance object, carefully consider its naming convention. Refer to System Data Naming Convention for more information.

Service Name is the name of the internal service associated with the maintenance object.

The grid displays the following for each table defined under the maintenance object:

- **Table** is the name of a given table maintained as part of the maintenance object.
- **Table Role** defines the table's place in the maintenance object hierarchy. Only one **Primary** table may be specified within a maintenance object, but the maintenance object may contain many **Child** tables.
- **Parent Constraint ID** specifies the constraint used to link the table to its parent table within the maintenance object table hierarchy.
- **Owner** indicates if this is owned by the base package or by your implementation (**Customer Modification**).

Maintenance Object - Options

Use this page to maintain a maintenance object's options. Open this page using **Admin** > **Database** > **Maintenance Object** and then navigate to the **Options** tab.

Description of Page

The options grid allows you to configure the maintenance object to support extensible options.

Select the **Option Type** drop-down to define its **Value**. **Detailed Description** may display additional information on the option type.

Set the **Sequence** to **1** unless the option can have more than one value.

Owner indicates if this is owned by the base package or by your implementation (**Customer Modification**).

> **Note: You can add new option types.** Your implementation may want to add additional maintenance option types. For example, your implementation may have plug-in driven logic that would benefit from a new type of option. To do that, add your new values to the customizable lookup field **MAINT_OBJ_OPT_FLG**.

Maintenance Object - Algorithms

Use this page to maintain a maintenance object's algorithms. Open this page using **Admin** > **Database** > **Maintenance Object** and then navigate to the **Algorithms** tab.

Description of Page

Copyright © 2024, Oracle and/or its affiliates.

The **Algorithms** grid contains algorithms that control important functions for instances of this maintenance object. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence Number** and **Algorithm** for each system event. You can set the **Sequence Number** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.
- If the algorithm is implemented as a script, a link to the **Script** is provided. Refer to Plug-in Scripts for more information.
- **Owner** indicates if this is owned by the base package or by your implementation (**Customer Modification**).

The following table describes each **System Event**.

System Event	Optional / Required	Description
Audit	Optional	Algorithms of this type are called to notify of any changes to the maintenance object's set of tables. These algorithms are invoked just before the commit at the end of a logical transaction. The system keeps track of what records are added or changed in the course of a transaction and all MO audit algorithms are executed in order of when each record was first added or updated.
Determine BO	Optional	Algorithm of this type is used to determine the Business Object associated with an instance of the maintenance object. It is necessary to plug in such an algorithm on a Maintenance Object to enable the business object rules functionality. The system invokes a single algorithm of this type. If more than one algorithm is plugged-in the system invokes the one with the greatest sequence number.
ILM Eligibility	Optional	Algorithms of this type are used for maintenance objects that are enabled for object erasure for Information Lifecycle Management. They are used to review records that have reached the maximum retention days and evaluate if they are ready to be archived. The system invokes a single algorithm of this type. If more than one algorithm is plugged-in the system invokes the one with the greatest sequence number.
Information	Optional	We use the term "Maintenance Object Information" to describe the basic information that appears throughout the system to describe an instance of the maintenance object. The data that appears in this information description is constructed using this algorithm. The system invokes a single algorithm of this type. If more than one algorithm is plugged-in the system invokes the one with the greatest sequence number.

Copyright © 2024, Oracle and/or its affiliates.

System Event	Optional / Required	Description
Manage Erasure Schedule	Optional	Algorithms of this type are used for maintenance objects that are enabled for object erasure, which is a measure to protect Data Privacy. They are triggered when certain system events occur for those objects. The algorithms are used to determine if the record needs to be scheduled for erasure and, if so, create or maintain an entry for the record in the Object Erasure Schedule.
Revision Control	Optional	An algorithm of this type is used to enforce revision control rules when an object is added, changed, or deleted. The maintenance object service calls the plug-in once before the object is processed and once more after applying all business object rules. This allows revision rules to take place in proper revision timings.
Transition	Optional	The system calls algorithms of this type upon each successful state transition of a business object as well as when it is first created. These are typically used to record the transition on the maintenance object's log. Note that most base maintenance objects are already shipped with an automatic logging of state transitions. In this case you may use these algorithms to override the base logging functionality with your own. Refer to State Transitions are Audited for more information.
Transition Error	Optional	The system calls this type of algorithm when a state transition fails and the business object should be saved in its latest successful state. The algorithm is responsible for logging the transition error somewhere, typically on the maintenance object's log. Notice that in this case, the caller does NOT get an error back but rather the call ends successfully and the exception is recorded somewhere, as per the plug-in logic. The system invokes a single algorithm of this type. If more than one algorithm is plugged-in the system invokes the one with the greatest sequence number.

Note: You can inactivate algorithms on Maintenance Objects. Your implementation may want to inactivate one or more algorithms plugged into the base maintenance object. To do that, go to the options grid on Maintenance Object - Options and add a new option, setting the option type to **Inactive Algorithm** and setting the option value to the algorithm code.

Copyright © 2024, Oracle and/or its affiliates.

Maintenance Object - Maintenance Object Tree

You can navigate to the **Maintenance Object Tree** to see an overview of the tables and table relationships associated with the maintenance objects.

Description of Page

This page is dedicated to a tree that shows the maintenance object's tables as well as business objects, if you have defined any. You can use this tree to both view high-level information about these objects and to transfer to the respective page in which an object is maintained.

Defining Lookups

Lookup fields may be used to define valid values for a column in a table or for other types of values like parameters to an algorithm. This portal is used to view and maintain information about a lookup field and its values. Note that all lookup information is also maintained as part of the corresponding Field portal.

You can access the portal from the**Admin Menu** > **Database** > **Lookup**. You are brought to a query portal with options for searching for a specific lookup field. Once a field has been selected you are brought to the maintenance portal to view and maintain the selected record.

Fastpath: Refer to Defining Valid Values for some background information.

Note: When introducing a new record, carefully consider its naming convention. Refer to System Data Naming Convention for more information.

The following zones may appear as part of the portal's **Main** tab page

- **Lookup.** This zone provides information about the selected lookup field.
- **Lookup Values.** This zone lists and maintains valid values for the selected lookup field.

The base product provides many different lookup fields and their values as part of the product. The following points highlight some functionality related to base-package lookups.

- Fields that are owned by the product will typically provide base lookup values. Implementations are not permitted to remove base delivered lookup values. Implementations may be able to add custom values to base owned lookups. This is controlled with the Custom switch on lookup.
  - When the custom switch is unchecked, it means that there is functionality controlled by the base values and an implementation may not extend or customize this functionality. An example of this type of lookup is the Data Type field on the Field table. The system supports a distinct list of data types and an implementation may not add additional values.
  - When the custom switch is checked, it means that there is base functionality supplied for the base values but that an implementation can extend the functionality by supplying their own values. An example of this type of lookup is the Access Mode on Application Service. The product provides many values for the access mode lookup, representing various actions a user may perform. Implementations may add their own values to this lookup. Documentation should indicate when functionality may be extended and should highlight the lookup value that can be extended.

    CAUTION: Important! If you introduce new lookup values, you must prefix the lookup value code with **X** or **Y**. If you do not do this, there is a possibility that a future release of the application could introduce a new lookup value with the name you allocated.
- There may be some scenarios where the product supplies a base field and base lookup field with no base lookup values supplied. This occurs when the product doesn't have any base functionality driven by the lookup values. Typically this type of lookup is for information or categorization purposes. The configuration guide for the functional area associated with the lookup should include a configuration step regarding defining values for this type of lookup.
- The description of base delivered values may be overridden by an implementation.

Copyright © 2024, Oracle and/or its affiliates.

An implementation may also identify the need for defining a new lookup field with its values.

Defining Extendable Lookups

Extendable lookups are a way of defining valid values that are more sophisticated than simple lookups.

Fastpath: Refer to Defining Valid Values for some background information.

The base product provides extendable lookups as part of the product. The following points highlight some functionality related to base-package extendable lookups.

- The base product may supply base extendable lookup values. Implementations are not permitted to remove base delivered extendable lookup values. It is also possible that implementations may be able to add custom values to base owned lookups. If an implementation is not permitted to add lookup values to the base extendable lookup, the extendable lookup's business object will include validation to prevent this. There is no equivalent of the Custom switch that is on the lookup field.
- There may be some scenarios where the product supplies a base extendable lookup with no base lookup values supplied. This occurs when the product doesn't have any base functionality driven by the extendable lookup values. The configuration guide for the functional area associated with the extendable lookup should include a configuration step regarding defining values for this type of extendable lookup.
- The description of base delivered values may be overridden by an implementation.

Open this page using **Admin** > **General** > **Extendable Lookup.**

You are brought to the **Extendable Lookup Query** where you need to search for the extendable lookup object (i.e., its business object).

Once you have found the appropriate extendable lookup, select the value and you are brought to a standard All-in-One portal that lists the existing lookup values for the extendable lookup. The standard actions for an All-in-One portal are available here.

Extendable Lookup Advanced Topics

This section provides some addition technical information about extendable lookup attributes

Defining Additional Attributes

The product provides a few different ways to define additional values for an extendable lookup. Some of the methods are only relevant for base delivered lookup values as they may impact whether or not an implementation can update the values.

The following table highlights the options available and some summary information about what the option provides.

Option	Brief Description	Extendable Lookup Value Searchable by this Attribute?	Base Delivered Value Modifiable?
Element mapped to BO_DATA_AREA	The element is mapped to a CLOB field that allows for base delivered values to be modified.	No	Yes
Element mapped to BASE_BO_DATA_AREA	The element is mapped to a CLOB field that does not allow for base delivered values to be modified.	No	No

Copyright © 2024, Oracle and/or its affiliates.

Option	Brief Description	Extendable Lookup Value Searchable by this Attribute?	Base Delivered Value Modifiable?
Flattened characteristic	The element is defined using the flattened characteristic mechanism.	Yes	No

The following points highlight information from the table above:

- The decision of defining an additional attribute using a CLOB mapping or a flattened characteristic will depend on whether the functionality expects that the lookup value is known when the attribute is needed (in which case a CLOB mapping is appropriate) or if the functionality expects to determine the lookup value based on the attribute (in which case, a flattened characteristic is appropriate).
- When the base product defines an extendable lookup with additional attributes and intends to provide base extendable lookup values, it needs to determine whether or not implementations may update the additional attribute or not.
  - If no and the value is mapped to a CLOB, it will map the value to the BASE_BO_DATA_AREA column. This means that implementations will receive an owner mismatch error when attempting to change the value. In addition, upgrading to a new release will replace the value with the base value.
  - If yes and the value is mapped to a CLOB, it will map the value to the BO_DATA_AREA column. This means that implementations will be able to change the value for a base owned record. In addition, upgrading to a new release will not make any changes to the value.
  - For values mapped to a characteristic, the product does not support an implementation changing the value of a base delivered record. If the product would like to support an implementation overriding this type of value, the business object will need to be designed with a corresponding "override" element (also a flattened characteristic), similar to how the product supplies an Override Description field to support an implementation overriding the base product delivered description for a base value. This element will not be delivered with any value and will allow an implementation to populate that value.

    Note: Note that in this situation, the product functionality that uses this value must cater for the override value.
- All of this detail is only relevant for base provided extendable lookup values. If an implementation adds custom values for a base supplied extendable lookup, all the additional attributes may be populated as appropriate.
- If an implementation defines a custom extendable lookup business object and wants to define an additional attribute using a CLOB, it doesn't matter which CLOB column is used. Both BO_DATA_AREA and BASE_BO_DATA_AREA provide the same functionality for custom business objects.

Capturing a Password

If an extendable lookup includes configuration of a password for some functionality, the system supports automatic encryption of the password value if the schema maps the password to a characteristic using the characteristic type **F1-PWD**.

Defining Characteristic Types

This portal is used to view and maintain characteristic types. An implementation may define their own characteristic types or override some of the settings for base product provided characteristic types.

You can access the portal from the**Admin Menu** > **General** > **Characteristic Type**. You are brought to a query portal with options for searching for a specific record. Once a characteristic type record has been selected you are brought to the maintenance portal to view and maintain the selected record.

Note: When introducing a new record, carefully consider its naming convention. Refer to System Data Naming Convention for more information.

Refer to Characteristic Types for more information.

Copyright © 2024, Oracle and/or its affiliates.

The following zones may appear as part of the portal's **Main** tab page

- **Characteristic Type.** This zone provides information about the selected characteristic type.
- **Characteristic Values.** This zone is visible for pre-defined characteristic types and shows the existing values (if applicable) with action buttons. It also allows you to add new values. Note that you may only add values to a product owned characteristic type if the Custom switch is checked.

Defining Foreign Key References

The topics in this section describe the transaction that allows you to define foreign key references.

Refer to Foreign Key References for more information.

To setup a foreign key reference, open **Admin** > **Database** > **FK Reference.**

Use the **Foreign Key Reference Query** portal to search for an existing foreign key reference. Once a foreign key reference is selected, you are brought to the maintenance portal to view and maintain the selected record.

CAUTION: Important! If you introduce a new foreign key reference, carefully consider its naming convention. Refer to System Data Naming Convention for more information.

Description of Page

Enter an easily recognizable **FK** (foreign key) **Reference** code and **Description** for the record.

Enter the name of the **Table** whose primary key is referenced. After selecting a **Table**, the columns in the table's primary key are displayed adjacent to **Table PK Sequence**.

Use **Navigation Option** to define the page to which the user will be transferred when they press the go to button or hyperlink associated with the referenced entity. Refer to Navigation Information Is Dynamically Derived for more information on how this is used.

The **Info Program Type** indicates whether the default program that returns the standard information description is **Java** or **Java (Converted)**, meaning it was converted into Java.

Note: Java (Converted) program types are not applicable to all products.

Use **Info Program Name** to enter the Java class / program name.

Refer to Information Description Is Dynamically Derived for more information on the info program is used.

View the source. If the program is shipped with the base package, you can use the adjacent button to display the source code of this program in the Javadocs viewer.

Use **Context Menu Name** to specify the context menu that appears to the left of the value.

Note: Context Menu Name is not applicable to user interface elements displaying a generic collection using a foreign key characteristic type. It is only applicable for pages utilizing the foreign key compound element type for fixed page user interface and for data displayed in a portal based user interface where the foreign key reference is defined as an attribute for an element. Report parameters that reference foreign key characteristics are an example of a user interface where a context menu is not displayed even if the foreign key reference defines one.

Use **Search Zone** to define the search zone that opens when a user searches for valid values when the foreign key reference is configured as an input field on a portal based page. Refer to Search Options for more information.

Use **Search Navigation Key** to define the search page that will be opened when a user searches for valid values on a user interface that is a fixed page. Refer to Search Options for more information.

Use **Search Type** to define the default set of search criteria used by the **Search Navigation Key**'s search page.

Use **Search Tooltip** to define a label that describes the **Search Navigation Key**'s search page.

Copyright © 2024, Oracle and/or its affiliates.

> **Note: Search Type and Search Tooltip**. These attributes are only applicable to user interface elements utilizing the foreign key compound element type on fixed page user interfaces. Report parameters that reference foreign key characteristics are an example of a user interface where this information is not used even if the foreign key reference defines them.

Audit Trails

The topics in this section describe one way of auditing changes in the system. Note that this technique has limitations and may not be the best option for all situations.

- This functionality is configured at the table and field level. Many base maintenance objects use an XML or CLOB field to capture one or more elements using XML format, configured using a BO schema. This auditing technique is not able to capture changes to individual elements. It can only capture overall changes to the single field. The BO Audit plug-in is a better option for auditing changes to individual elements with the CLOB / XML field.
- The base table provided for supporting audits limits the field size of the before and after values to 254 bytes.
- Auditing is captured for each field. In some cases it is prefered to capture a before and after image for several fields at once. For example, if an address is changed, it’s nore user friendly to capture the before and after for the full address rather than the individual address components. The BO Audit plug-in allows for capturing a single audit record for multiple elements rather than granular changes for each element.

The subsequent topics highlight how to enable auditing for fields, and describe the auditing queries that you can use to view audit records.

Captured Information

When auditing is enabled for a field, the following information is recorded when the field is changed, added and/or deleted (depending on the actions that you are auditing for that field):

- User ID
- Date and time
- Table name
- Row's prime key value
- Field name
- Before image (blank when a row is added)
- After image (blank when a row is deleted)
- Row action (add, change, delete)

How Auditing Works

You enable auditing on a table in the table's meta-data by specifying the name of the table in which to insert the audit information (the audit table) and the name of the program responsible for inserting the data (the audit trail insert program). Then you define the fields you want to audit by turning on each field's audit switch in the table's field meta-data. You can audit fields for delete, insert and update actions.

Once auditing is enabled for fields in a table, the respective row maintenance program for the table assembles the list of changed fields and calls the audit trail insert program. If any of the changed fields are marked for audit, the audit program inserts audit rows into the audit table.

> **Note: Customizing Audit Information.** You may want to maintain audit information other than what is described in Captured Information or you may want to maintain it in a different format. For example, you may want to maintain audit information for an entire row instead of a field. If so, your implementation team can use the base audit program and base audit tables as examples when creating your own audit trail insert program and audit table structures.

Copyright © 2024, Oracle and/or its affiliates.

The Audit Trail File

Audit log records are inserted in the audit tables you define. The base product contains a single such table (called **CI_AUDIT**). However, the base audit insert program is designed to allow you to use multiple audit tables.

If you want to segregate audit information into multiple tables, you must create these tables. Use the following guidelines when creating new audit tables (that use the base delivered audit insert program):

- The new audit tables must look identical to the base table (**CI_AUDIT**).
- The new tables must be prefixed with **CM** (e.g., **CM_AUDIT_A**, **CM_AUDIT_B**, etc.). An appropriate java table name must be defined and an *_impl.java program for that java table name must be defined.
- The name of the new table must be referenced on the various tables whose changes should be logged in the new table.

Note: It's important to note if you use your own tables (as opposed to using the base package table called **CI_AUDIT**), the SQL used to insert and access audit trail records in the base delivered audit program is dynamic. Otherwise, if the base package's table is used, the SQL is static.

How To Enable Auditing

Enabling audits is a two-step process:

- First, you must turn on auditing for a table by specifying an audit table and an audit trail insert program.
- Second, you must specify the fields and actions to be audited for the table.

The following topics describe this process.

Turn On Auditing For a Table

In order to tell the system which fields to audit, you must know the name of the table on which the field is located. You must specify the audit table and the audit trail insert program for a table in the table's meta-data.

Note: Most of the system's table names are fairly intuitive. For example, the user table is called SC_USER, the navigation option table is called CI_NAV_OPT, etc. If you cannot find the table using the search facility on the Table portal, please contact customer support.

To enable auditing for a table:

- Navigate to the Table portal and find the table associated with the field(s) for which you want to capture audit information.
- Specify the name of the **Audit Table**.

Note: Specifying the Audit Table. You can use the audit table that comes supplied with the base package (**CI_AUDIT**) to audit multiple tables and fields. All the audit logs are combined in a single table (**CI_AUDIT**). However, you can also have a separate audit table for each audited table. Refer to The Audit Trail File for more information.

- Specify the name of the **Audit Program** . The product provides two programs. **com.splwg.base.domain.common.audit.DefaultTableAuditor** is a standard table audit program. In addition **com.splwg.base.domain.common.audit.ModifiedTableAuditor**is supplied as an alternative. This program does not audit inserts or deletes of empty string data.

CAUTION: By default, none of a table's fields are marked for audit. Even though you have enabled auditing for a table, you must still specify the fields and actions on those fields to be audited (see below).

Specify The Fields and Actions To Be Audited

The system only audits actions (insert, update and delete) made to fields that you want audited.

To specify the fields and actions to be audited:

Copyright © 2024, Oracle and/or its affiliates.

- Navigate to the Table portal for a table on which you have enabled auditing.
- For each field you want to audit, specify the actions you want to audit by turning on the **Audit Delete**, **Audit Insert** and **Audit Update** switches as appropriate.

Note: You can also turn on the audit switches using the Field zone at the bottom of the Table portal.

CAUTION: Audit Program Caching! The audit program from the table meta-data is read into a program cache on the application server whenever the date changes or when the server starts. If you implement new auditing on a table, your audit trail does not become effective until this program cache is reloaded. In other words, new audits on tables where the audit program was not previously specified do not become effective until the next day (or the next restart of the application server). However, if you change the fields to be audited for a table where the audit program is already in the cache, your changes are effective immediately.

Audit Queries

There are two queries that can be used to access the audit information.

Audit Query by User

This transaction is used to view changes made by a user that are stored on a given Audit Trail File.

CAUTION: The system only audits changes that you've told it to audit. Refer to The Big Picture Of Audit Trails for more information.

Navigate to this page by selecting **Admin** > **Database** > **Audit Query By User**.

Description of Page

To use this transaction:

- Enter the **User ID** of the user whose changes you wish to view.
- Enter the name of the table on which the audit trail information is stored in **Audit Table**. Refer to The Audit Trail File for more information about this field.

Note: Default Note. If only one audit table is used to store audit trail information, that table is defaulted.

- Specify a date and time range in **Created between** to restrict the records that result from the query.

Note: Default Note. The current date is defaulted.

- Click the search button to display all changes recorded on a specific audit table associated with a given user.

Information on this query is initially displayed in reverse chronological order.

The following information is displayed in the grid:

- **Row Creation Date** is the date and time that the change was made.
- **Audited Table Name** contains the name of the table whose contents were changed.
- **Primary Key** is the prime key of the row in the **Audited Table** whose contents where changed.
- **Audited Field Name** is the name of the field that was changed.
- **Audit Action** indicates whether the row action was **Add**, **Change** or **Delete**.
- **Field Value Before Update** contains the content of the field before the change. This column is blank if information was **Added**.
- **Field Value After Update** contains the content of the field after the change. This column is blank if information was **Deleted**.

Copyright © 2024, Oracle and/or its affiliates.

Audit Query by Table / Field / Key

This transaction is used to view audited changes made to a given table.

> **CAUTION:** The system only audits changes that you've told it to audit. Refer to The Big Picture Of Audit Trails for more information.

This transaction can be used in several different ways:

- You can view all audited changes to a table. To do this, enter the **Audited Table Name** and leave the other input fields blank.
- You can view all audited changes to a given row in a table (e.g., all changes made to a given user). To do this, enter the **Audited Table Name** and row's prime key (the row's prime key is entered in the field(s) beneath **Audited Field Name**).
- You can view all audited changes to a given field in a table (e.g., all changes made to all customers' rates). To do this, enter the **Audited Table Name** and the **Audited Field Name**.
- You can view all audited changes to a given field on a specific row. To do this, enter the **Audited Table Name**, the **Audited Field Name**, and row's prime key (the row's prime key is entered in the field(s) beneath **Audited Field Name**).

Navigate to this page by selecting **Admin** > **Database** > **Audit Query By Table/Field/Key**.

Description of Page

To use this transaction:

- Enter the name of the table whose changes you wish to view in **Audited Table Name**.
- If you wish to restrict the audit trail to changes made to a specific field, enter the **Audited Field Name**.
- If you wish to restrict the audit trail to changes made to a given row, enter the row's prime key (the row's prime key is entered in the field(s) beneath **Audited Field Name**). These fields are dynamic based on the **Audited Table Name**.
- Specify a date and time range in **Created between** to restrict the records that result from the query.

> **Note:** The current date is defaulted.

- Click the search button to display all changes made to this data.

Information on this query is initially displayed in reverse chronological order by field.

The following information is displayed in the grid:

- **Create Date/Time** is the date / time that the change was made.
- **User Name** is the name of the person who changed the information.
- **Primary Key** is the prime key of the row in the **Audited Table** whose contents where changed.
- **Audited Field Name** is the name of the field that was changed.
- **Audit Action** indicates whether the row action was **Add**, **Change** or **Delete**.
- **Value Before Update** contains the content of the field before the change. This column is blank if information was **Added**.
- **Value After Update** contains the content of the field after the change. This column is blank if information was **Deleted**.

Bundling

The topics in this section describe the bundling features in the application.

About Bundling

Bundling is the process of grouping entities for export or import from one environment to another.

Copyright © 2024, Oracle and/or its affiliates.

For example, you might export a set of business objects and service scripts from a development environment and import them into a QA environment for testing. The group of entities is referred to as a bundle. You create export bundles in the source environment; you create import bundles in the target environment.

Working with bundles involves the following tasks:

- Configuring entities for bundling if they are not preconfigured
- Creating an export bundle, which contains a list of entities to be exported from the source environment
- Creating an import bundle to import those entities to the target environment
- Applying the import bundle, which adds or updates the bundled entities to the target environment

Sequencing of Objects in a Bundle

Bundle entities are added or updated to the target environment in the sequence defined in the bundle

Typically, the sequence of entities does not matter. However, sequence is important in the following situations:

- Entities that are referenced as foreign keys should be at the top of the sequence, before the entities that reference them. Specify zones last, as they typically contain numerous foreign key references.
- When importing a business object, specify the business object first, then its plug-in scripts, then the algorithms that reference the scripts, and then the algorithm types that reference the algorithms.
- When importing a portal and its zones, specify the portal first and then its zones.
- When importing a multi-query zone, specify the referenced zones first and then the multi-query zone.
- Always specify algorithms types before algorithms.

You can specify the sequence when you define the export bundle or when you import the bundle to the target environment.

Recursive Key References

Recursive foreign keys result when one object has a foreign key reference to another object that in turn has a foreign key reference to the first object.

For example, a zone has foreign keys to its portals, which have foreign keys to their zones. If the objects you want to bundle have recursive relationships, you must create a 'bundling add' business object that has only the minimal number of elements needed to add the entity. A bundling add business object for a zone contains only the zone code and description, with no references to its portals. In the same way, a bundling add business object for a portal defines only its code and description.

When you apply the bundle, the system initially adds the maintenance object based on the elements defined in the bundling add business object. Before committing the bundle, the system updates the maintenance object with the complete set of elements based on its physical business object.

Note that use of the bundling add BO also benefits records that have optional foreign keys or foreign keys that are part of a child table. That way the person creating the bundle does not have to worry about the sequence of the records. For example, an FK Reference may optionally reference a zone for searching. If a new FK Reference and its search zone are bundled together to copy to another region, the bundling add BO for FK Reference (which doesn’t include the Zone) ensures that the FK reference could be added before the zone without getting any validation errors.

Owner Flags on Bundled Entities

The owner flag of the entities in an import bundle must match the owner flag of the target environment.

If you need to import objects that your source environment does not own, you must replace the owner flag in the import bundle with the owner flag of the target environment.

Copyright © 2024, Oracle and/or its affiliates.

Configuring Maintenance Objects for Bundling

Most base package meta-data objects are pre-configured to support bundling. All other objects must be manually configured.

If a base package maintenance object is pre-configured for bundling, the **Eligible For Bundling** option will be set to "Y" on the Options tab for the maintenance object.

To configure other objects for bundling, review the configuration tasks below and complete all those that apply:

Configuration Task	Scope of Task
Make maintenance objects eligible for bundling	All objects to be included in the bundle.
Add a foreign key reference	All objects to be included in the bundle.
Create a physical business object	All objects to be included in the bundle.
Create a bundling add business object	Only needed if there are objects with recursive or optional foreign key references.
Add the Current Bundle zone	Perform this step for any object where you want the Current Bundle zone to appear on the sidebar when on the maintenance page for this maintenance object. This is not required by the bundling process.
Create a custom Entity Search zone and add it to the Bundle Export portal	Perform this step if you would like the object to be listed in the Entity Search zone in the Bundle Export portal. This is not required by the bundling process.

Making Maintenance Objects Eligible for Bundling

The "Eligible For Bundling" maintenance object option must be set to "Y" for all bundled objects. Go to the Maintenance Object page and search for the maintenance object. On the Options tab, add a new option with the type **Eligible For Bundling**. Set the value to "Y" and click **Save**. This configuration must be in both the source environment and the target environment.

Adding a Foreign Key Reference

Foreign key references are used for many purposes and most maintenance objects should have them anyway. Check the Maintenance Object options to see if there is an entry for the **Foreign Key Reference** option. If one is not configured, navigate to FK Reference and set up a foreign key reference for the primary table of the maintenance object. Configure that new FK reference on the maintenance object using an option.

Creating a Physical Business Object

If the maintenance object does not already have a Physical BO configured, one must be created to support using it in a bundle. Refer to Creating a Physical Business Object for more information.

Creating a Bundling Add Business Object

If the objects to be bundled have recursive foreign key references, optional foreign key references or child tables that include foreign key references, create a bundling add business object to avoid problems with referential integrity.

1. Navigate to Business Object and add a business object for the maintenance object. Do not fill in any of the UI oriented configuration.
2. The business object is created with a schema that looks like the physical structure of the maintenance object.
3. Remove all child tables and all elements that are not required.
4. Save the business object.

Copyright © 2024, Oracle and/or its affiliates.

5. Navigate to Maintenance Object and search for the maintenance object you want to bundle.
6. On the **Option** tab, add a new option with the type **Bundling Add BO**. The value is the name of the bundling add business object you just created.

Adding the Current Bundle Zone

If you want the Current Bundle zone to appear on the sidebar when the a user is on the maintenance page for the maintenance object, you must add the Current Bundle zone as a context-sensitive zone for the maintenance page's navigation key.

1. Navigate to Context Sensitive Zone and search for the navigation key for the maintenance page of this object.
2. Add the Current Bundle zone F1-BNDLCTXT, to that navigation key.

Adding a Customized Entity Search Query Zone to the Bundle Export Portal

If you want the maintenance object to be searchable in the Bundle Export portal in the Entity Search zone, you must first create an entity-specific query zone to search for the maintenance object. Then you must create a customized entity search zone that references this new query zone. Finally, you must add the customized entity search zone to the Bundle Export portal.

Note that not all base product entities that are eligible for bundling have a search configured for this zone. There is a limit to the number of searches that may be listed here. The base product has provided searches for only the most common objects that are bundled.

Create an entity-specific query zone to search for the maintenance object:

1. Navigate to Zone and search for one of the base query zones, such as the Algorithm Search zone F1-BNALGS.
2. Click the **Duplicate** button in the record actions section.
3. Enter a name and description for the new zone.
4. Click **Save**.
5. Customize the details of the parameters for your maintenance object.

Create a customized entity search zone. This step only needs to be done once. If you already have a customized search zone in the Bundle Export portal, skip this step.

1. Navigate to Zone and search for the F1-BNDLENTQ Entity Search zone.
2. Duplicate this zone.
3. Remove any references to base query zones.

Add the new entity-specific query zone to the customized entity search zone:

1. Locate the customized entity search zone for your Bundle Export portal. This is the zone created in the previous step.
2. Add the name of the query zone you created in the first step to the list of Query Zones.

Add the customized entity search zone to the Bundle Export portal. This step needs to be done only once.

1. Navigate to Portal and search for the Bundle Export portal, F1BNDLEM.
2. In the zone list, add the entity search zone you created above. (Add the new zone after the base entity search zone).
3. Save the portal.

Working with Bundles

Use the Bundle Export portal to create an export bundle. The export bundle contains a list of entities to be exported from the source environment. When you are ready to import the objects, use the Bundle Import portal to import the objects to the target environment.

Creating Export Bundles

An export bundle contains a list of entities that can be imported into another environment.

Copyright © 2024, Oracle and/or its affiliates.

To create an export bundle:

1. Log on to the source environment from which objects will be exported.
2. Select **Admin** > **Implementation Tools** > **Bundle Export** > **Add**.
3. Complete the fields in the Main section to define the bundle's basic properties.

   Note: You can use the Entities section to add bundle entities now, or save the bundle and then add entities as described in step 5.

4. Click **Save** to exit the Edit dialog. The export bundle status is set to Pending.
5. While an export bundle is in Pending state, use any of the following methods to add entities to the bundle:
   a) Use the **Entity Search** zone on the Bundle Export portal to search for entities and add them to the bundle. If an entity is already in the bundle, you can remove it.
   b) To import entities from a .CSV file, click **Edit** on the Bundle Export portal, and then click **CSV File to Upload.** Specify the file name and location of the .CSV file containing the list of entities. Click **Submit** to upload the file, and then click **Save** to save the changes.
   c) Use the **Current Bundle** zone in the sidebar of the entity you want to add. (All entities that are configured to support bundling display a Current Bundle zone). This zone displays a list of all pending export bundles to which you can add the entity.
   d) When you check an entity into revision control, specify the export bundle on the **Revision Info** dialog.
6. When you have added all entities, click **Bundle** in the Bundle Actions zone on the Bundle Export portal. The export bundle state is set to Bundled and the Bundle Details zone displays the XML representation of every entity in the bundle.

   Note: The owner flags of the entities in the bundle must match the owner flag of the bundle itself. If the owner flags do not match, the system displays a warning message. Click **OK** to continue or **Cancel** to abort the bundle. If you click OK, you will need to resolve the owner flag discrepancy before you import the bundle to the target environment.

7. Copy the XML from the **Bundle Detail** zone to the clipboard (or to a text file). You can now create an import bundle and apply it to the target environment.

   Note: If you need to make additional changes to the bundle, you must change the bundle state by selecting the **Back to Pending** button in the **Bundle Actions** zone.

Creating and Applying Import Bundles

Import bundles define a group of entities to be added or updated in the target environment.

Before you create an import bundle, you must have already created an export bundle, added entities, and set the bundle's state to Bundled.

To create an import bundle and apply it to the target environment:

1. If you have not already copied the XML from the export bundle, do so now:
   a) Select **Admin** > **Implementation Tools** > **Bundle Export** and search for the bundle.
   b) Copy the XML from the **Bundle Detail** zone to the clipboard (or to a text file).
2. Log on to the target environment.
3. Select **Admin** > **Implementation Tools** > **Bundle Import** > **Add**.
4. In the **Bundle Actions** zone, click **Edit XML**.
5. Paste the contents of the clipboard (or text file if you created one) into the **Bundle Detail** zone.
6. Make any necessary changes to the XML and click **Save**. The status of the import bundle is set to Pending.

   Note: Use caution when editing the XML to avoid validation errors.

7. To remove entities from the import bundle or change their sequence, click **Edit**. Enter your changes and click **Save** to exit the Edit dialog.

Copyright © 2024, Oracle and/or its affiliates.

8. When you are ready to apply the bundle, click **Apply**. The import bundle state is set to Applied and the entities are added or updated in the target environment.

Editing Export Bundles

You can add or remove entities from an export bundle when it is in Pending state. You can also change the sequence of entities.

To edit to an export bundle that has already been bundled, you must change the bundle state by selecting the **Back to Pending** button on the Bundle Export portal.

To edit a pending export bundle:

1. Open the bundle in edit mode.
2. Click **Edit** on the Export Bundle portal.
3. Make any necessary changes on the edit dialog and then click **Save**.

Editing Import Bundles

You can remove entities from an import when it is in Pending state. You can also change the sequence of entities and edit the generated XML.

To edit a pending import bundle:

1. Open the bundle in edit mode.
2. To edit the XML snapshot, click **Edit XML**. Edit the XML code as needed, then click **Save**.

   Note: Use caution when editing the XML to avoid validation errors.

3. To remove entities or change their sequence, click **Edit**. Make any necessary changes and click **Save**.

Revision Control

The topics in this section describe the revision control features in the application.

About Revision Control

Revision control is a tool provided for the development phase of a project to allow a user to check out an object that is being worked on. In addition, it captures the version of the maintenance object when users check in an update, maintaining a history of the changes to the object.

If revision control is enabled for an object you must check out the object to change it. While the object is checked out no one else can work on it. You can revert all changes made since checking out an object, reinstate an older version of an object, recover a deleted object, and force a check in of an object if someone else has it checked out.

Note: Revision control does not keep your work separate from the environment. Because the metadata for maintenance objects is in the central database, any changes you make to an object while it is checked out will be visible to others and may impact their work.

Many of the maintenance objects used as configuration tools are already configured for revision control, but it is turned off by default. For example, business objects, algorithms, data areas, UI maps, and scripts are pre-configured for revision control.

Turning On Revision Control

Revision control is turned off by default for maintenance objects that are configured for revision control.

To turn on revision control:

Copyright © 2024, Oracle and/or its affiliates.

1. Add the base package **Checked Out** zone to the Sidebar portal.
   a) Navigate to Portal.
   b) Search for the portal CI_DASHBOARD.
   c) In the zone list for the **Sidebar** portal, add the zone F1-USRCHKOUT.
2. Set up application security.

   For users to have access to revision control, they must belong to a user group that has access to the application service F1-OBJREVBOAS.
3. Add the revision control algorithm to the maintenance object that you want to have revision control.

   This step must be repeated for each maintenance object that you want to have revision control.
   a) Go to the Maintenance Object page and search for the maintenance object that you want to have revision control.
   b) On the **Algorithms** tab of the maintenance object, add the revision control algorithm F1-REVCTL.

Configuring Maintenance Objects for Revision Control

Most configuration tool maintenance objects are pre-configured for revision control. You can configure other maintenance objects for revision control, as well.

To configure other objects for revision control:

1. Create a physical business object for the maintenance object. Refer to Creating a Physical Business Object for more information.
2. Add a foreign key reference to the maintenance object so that the standard information string is displayed. Foreign key references are used for many purposes and most maintenance objects should have them anyway. Check the Maintenance Object options to see if there is an entry for the **Foreign Key Reference** option. If one is not configured, navigate to FK Reference and set up a foreign key reference for the primary table of the maintenance object. Configure that new FK reference on the maintenance object using an option.
3. Add the **Revision Control** zone to the maintenance object.
   a) Navigate to Context Sensitive Zone and search for the navigation key for the maintenance object.
   b) Add the Revision Control zone, F1-OBJREVCTL, to that navigation key.
4. Add the revision control algorithm to the maintenance object.
   a) Go to the Maintenance Object page and search for the maintenance object that you want to have revision control.
   b) On the **Algorithms** tab of the maintenance object, add the revision control algorithm F1-REVCTL.

Working with the Revision Control Zones

You use two zones in the sidebar to work with revision controlled objects when revision control is turned on.

The **Revision Control** zone gives you several options for managing the revision of the currently displayed object. This zone also shows when the object was last revised and by whom. This information is linked to the **Revision Control Search** portal which lists all of the versions of the object.

Using the Revision Control zone you can:

- Check out an object in order to change it.
- Check in an object so others will be able to work on it.
- Revert the object back to where it was at the last checkout.
- Force a check in of an object that is checked out by someone else. You need special access rights to force a check in.
- Delete an object.

The **Checked Out** zone lists all of the objects that you currently have checked out. Clicking on an object listed in this zone will take you to the page for that object. The zone is collapsed if you have no objects checked out.

See Revision Control Search for more information about Check In, Force Check In, and Check Out one or more records simultaneously.

Copyright © 2024, Oracle and/or its affiliates.

Checking Out an Object

You must check out a revision controlled object in order to change it.

An object must have revision control turned on before you can check it out.

> **Note:** When you first create or update an object a dialog box informs you that the object is under revision control. You can select **OK** to check out the object and save your changes, or **Cancel** to stop the update.

1. Go to the object that you want to work on.
2. Select **Check Out** in the **Revision Control** sidebar zone.

Checking In an Object

You must check in a revision controlled object in order to create a new version of it. Checking in an object also allows others to check it out.

1. Select a link in the **Checked Out** sidebar zone to go to the object that you want to check in.
2. Select **Check In** in the **Revision Control** sidebar zone.
3. Provide details about the version:
   - In the **External References** field state the bug number, enhancement number, or a reason for the revision.
   - In the **Detailed Description** field provide additional details regarding the revision.
   - In the **Keep Checked Out** box specify if you want to keep the object checked out. If you keep the object checked out then your revision is a new version that you can restore later.
   - In the **Add To Bundle** box specify if the object belongs to a bundle.
4. Select **OK** to check in the object.

Reverting Changes

Reverting changes will undo any changes you made since you checked out an object.

To revert changes:

1. Go to the object that you want to revert.
2. Select **Revert** in the **Revision Control** sidebar zone.
3. In the confirmation dialog box select **OK** to confirm the action or **Cancel** to return to the object page.

Once reverted, the object can be checked out by another user.

Forcing a Check In or Restore

You can force a check in if an object is checked out by another user and that person is not available to check it in.

You must have proper access rights to force a check in or restore.

To force a check in or restore:

1. Go to the object that is checked out by another user.
2. Select **Force Check In** or **Force Restore** in the Revision Control zone.

The **Force Check In** option is the same as a regular check in. The **Force Restore** option checks in the object and restores it to the previously checked in version.

Deleting an Object

If revision control is turned on for an object, you must use the **Revision Control** zone to delete it.

The object must be checked in before it can be deleted.

To delete a revision controlled object:

Copyright © 2024, Oracle and/or its affiliates.

1. Go to the object that you want to delete.
2. Select **Delete** in the **Revision Control** zone.
3. Provide details regarding the deletion.
4. Select **OK** to delete the object.

The system creates a revision record before the object is deleted so that the deleted object can be restored.

Restoring an Object

You can restore an older version of either a current object or a deleted object.

An object must be checked in before an older version can be restored.

To restore an object:

1. Go to the **Revision History** portal for the object.
   If the object was deleted you must search for it by going to **Admin** > **Implementation Tools** > **Revision Control**.
2. Select the desired entity by clicking the hyperlink in the **Details** column.
3. Locate the row in the version history that has the version that you want to restore and click **Restore**.
4. In the confirmation dialog box select **OK** to confirm the action or **Cancel** to return to the object page.

Working with the Revision Control Portal

The **Revision Control** portal lists information about each version of a revision controlled object.

You can navigate to the **Revision Control** portal from either a link in the **Revision Control** sidebar zone or by going to **Revision Control** portal through **Admin**.

If you want to find the Revision History entry for an earlier version or deleted object, you must search for the object using the **Revision Control Search** portal. Once you select the desired entry, you can restore a previous version of the object clicking **Restore** in the row for the version that you want to restore. You can also see the details of each version by clicking the broadcast icon for that version.

See Working with Revision Control Zones for more information about tasks that can be performed through Revision Control.

Revision Control Search

The **Revision Control Search** portal allows users to search for entities that have a revision history. The **Search By** dropdown provides additional functionality so that users can search for revisions that are associated to theirs or other's user ID. Users can also **Check In**, **Force Check In**, or **Check Out** one or more entities through this portal.

Zone Options

- **Revision History Search** allows the user to query for revised entities based on a combination of criteria.
  - In the **User ID** field, enter the user ID that is associated with a revision.
  - In the **External Reference** field, enter an ID from an external system and is associated with a revision.
  - In the **Maintenance Object** dropdown menu, select the Maintenance Object that is associated with a revision. The options in this list are populated by the Maintenance Objects that are active to track revision.
  - In the **Key 1**, **Key 2**, **Key 3**, **Key 4**, **Key 5** fields, enter the primary identifier(s) for the revised entity. Typically, the entity only requires a single key, but some entities require more than one (for example, Oracle Utilities Customer Care and Billing SA Type require CIS Division and SA Type).
  - In the **Status** dropdown menu, select the entity status for your search.
- **Check In** allows the user to search for entities currently checked out to the logged in user ID and a combination of criteria. Once the search results are returned, the user has the option to select one or more entities and check them in.

Copyright © 2024, Oracle and/or its affiliates.

  - In the **Maintenance Object** dropdown menu, select the Maintenance Object that is associated with a revision. The options in this list are populated by the Maintenance Objects that are active to track revision.
  - In the **Key** field, enter the primary identifier(s) for the revised entity.
- **Force Check In** allows the user to search for entities that are currently checked out by other user IDs (excluding the logged in user ID) based on a combination of criteria. Once the search results are returned, the user has the option to select one or more entities and check them in.
  - In the **Checked Out By User** field, enter the user ID that has the entity in a Checked Out status.
  - In the **Maintenance Object** dropdown menu, select the Maintenance Object that is associated with a revision. The options in this list are populated by the Maintenance Objects that are active to track revision.
  - In the **Key** field, enter the primary identifier(s) for the revised entity.
- **Check Out** allows the user to search for entities currently checked in user ID and a combination of criteria. Once the search results are returned, the user has the option to select one or more entities and check them out.
  - In the **Maintenance Object** dropdown menu, select the Maintenance Object that is associated with a revision. The options in this list are populated by the Maintenance Objects that are active to track revision.
  - In the **Key** field, enter the primary identifier(s) for the revised entity.

  Please see Working with Revision Control Zones for more information about working with individual entities.

Information Lifecycle Management

Information Lifecycle Management (ILM) is designed to address data management issues, with a combination of processes and policies so that the appropriate solution can be applied to each phase of the data's lifecycle.

Data lifecycle typically refers to the fact that the most recent data is active in the system. As time progresses, the same data becomes old and unused. Older data becomes overhead to the application not only in terms of storage, but also in terms of performance. This older data's impact can be reduced by using advanced compression techniques, and can be put into slower and cheaper storage media. Depending on how often it's accessed, it can be removed from the system to make an overall savings of cost and performance. The target tables for ILM are transactional tables that have the potential to grow and become voluminous over time.

The Approach to Implementing Information Lifecycle Management

This section describes the product approach to implementing ILM for its maintenance objects (MOs).

Note: The term archiving is used to cover any of the possible steps an implementation may take in their data management strategy, including compression, moving to cheaper storage, and removing the data altogether.

Age is the starting point of the ILM product implementation for some of its high volume data. In general "old" records are considered eligible to be archived. In the product solution, maintenance objects (MOs) that are enabled for ILM have an ILM Date on the primary table and the date is typically set to the record's creation date. (An MO may have special business rules for setting this date, in which case, a different date may be used to set the initial ILM Date). For implementations that want to use ILM to manage the records in the MO, the ILM date is used for defining partitions for the primary table.

There are cases where a record's age is not the only factor in determining whether or not it is eligible to be archived. There may be some MOs where an old record is still 'in progress' or 'active' and should not be archived. There may be other MOs where certain records should never be archived. To evaluate archive eligibility using information other than the ILM Date, the ILM enabled MOs include an ILM Archive switch that is used to explicitly mark records that have been evaluated and should be archived. This allows DBAs to monitor partitions based on age and the value of this switch to evaluate data that may be ready to be archived.

Evaluating records to determine their archive eligibility should still occur on "old" records. The expectation is that a large percentage of the old records will be eligible for archiving. The small number that may be ineligible could be

Copyright © 2024, Oracle and/or its affiliates.

updated with a more recent ILM date. This may cause the records to move into a different partition and can delay any further evaluation of those records until more time has passed.

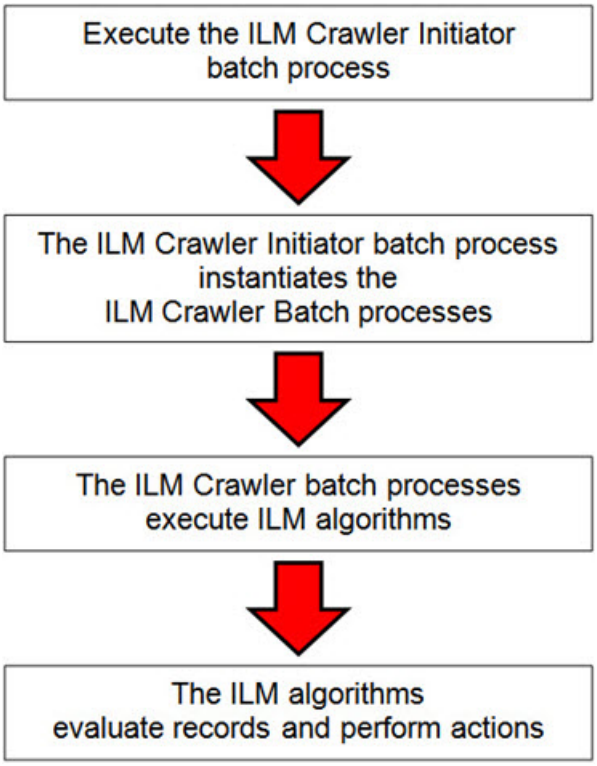
For each MO enabled for ILM, the product provides a batch process to review “old” records and an ILM eligibility algorithm that contains business logic to evaluate the record and mark it eligible for archiving or not. The following sections provide more information about the batch process and algorithm functionality.

Batch Processes

There are two main types of batch processes that manage data for ILM in the application: ILM Crawler Initiator and individual ILM Crawlers (one for each MO that is configured for ILM).

- **ILM Crawler Initiator**: (**F1-ILMIN**) - The ILM Crawler Initiator is a *driver* batch process that starts the individual ILM Crawler batch control as defined by the MO’s options.

  Restartable: In case of server failure, the ILM Crawler Initiator process can be restarted, which will also restart the ILM Crawler processes.
- **ILM Crawler**: Each maintenance object that is configured for ILM defines an ILM Crawler. These are *child* batch processes that can be started either by the ILM Crawler Initiator or by a standalone batch submission.



The ILM Crawler batch process selects records whose retention period has elapsed and invokes the MO's ILM eligibility algorithm to determine if the record is ready to be archived or not. The ILM eligibility algorithm is responsible for setting the record's ILM archive switch to 'Y' and updating the ILM date, if necessary.

The retention period defines the period that records are considered active. It spans the system date and cutoff date (calculated as system date - retention days).

The retention days of an MO is derived as follows:

- If the ILM Retention Days MO option is defined, that is used.
- Otherwise, the Default Retention Days from the ILM Master Configuration record is used.

An error is issued if no retention period is found.

Copyright © 2024, Oracle and/or its affiliates.

The crawler calculates the cutoff date and selects all records whose ILM archive switch is 'N' and whose ILM date is prior to the cutoff date. Each record returned is subject to ILM eligibility.

If the Override Cutoff Date parameter is supplied, it will be used instead of the calculated cutoff date. An error is issued if the override cutoff date is later than the calculated cutoff date. This parameter is useful if an object has many years of historic data eligible for archiving. Setting this parameter allows for widening the retention period and therefore limiting the process to a shorter period for initial processing.

Note: ILM Crawler batch processes are designed not to interfere with current online or batch processing. Because of this, these batch processes can run throughout the day.

Note: Before passing the cut-off date to the algorithm, the ILM crawler ensures that the number of days calculated (System Date – override cut-off date) is more than the retention period specified in the MO option or the Master Configuration. If the number of days calculated **is less than** the retention period specified on the MO option or the Master Configuration, then it throws an error.

Eligibility Algorithm

Algorithms are triggered by the ILM batch crawler for the maintenance object. The key responsibility of the ILM algorithm is to determine whether a record can be marked as ready to be archived or not. If a record is determined to be ready for archive, the algorithm should set the ILM Archive switch to Y. If not, the algorithm leaves the switch set to N and may decide to update the ILM Date to something more recent (like the System Date) to ensure that the record does not get evaluated again until the future.

This algorithm is plugged into the Maintenance Object - Algorithm collection.

Oracle Utilities Application Framework provides the algorithm **ILM Eligibility Based on Status** (**F1-ILMELIG**) to support the ILM batch crawler. Refer to the algorithm type description for details about how this algorithm works. If a maintenance object has special business rules that are evaluated to determine the eligibility for ILM, a custom algorithm can be created and applied by the implementation team.

Enabling ILM for Supported Maintenance Objects

In order to enable ILM for one or more maintenance objects, several steps are needed in both the configuration and in the database. This section describes some configuration enabled by default and some steps that must be taken in order to fully implement ILM.

There is some configuration enabled by default, but it will not be used unless ILM is fully configured. Each maintenance object that the product has configured for ILM has the following provided out of the box:

- **Special Table Columns**: Maintenance objects that support ILM include two specific columns: ILM Archive Switch (**ILM_ARCH_SW**) and ILM Date (**ILM_DT**).
- **Crawler Batch Process**: A "crawler" batch process is provided for each maintenance object that supports ILM and it is plugged into the MO as an option. Refer to Batch Processes for more information.
- **ILM Eligibility Algorithm**: Each maintenance object that is configured for ILM defines an eligibility algorithm that executes the logic to set the ILM Archive switch appropriately. This is plugged in to the MO algorithm collection.

If an implementation decides to implement ILM, there are steps that need to be followed, which are highlighted below.

Create the Master Configuration Record

The first step when enabling ILM is to create the **ILM Configuration** master configuration record.

The master configuration for ILM Configuration defines global parameters for all ILM eligible maintenance objects. For example, the Default Retention Period. In addition, your product may implement additional configuration. Refer to the embedded help for specific details about the information supported for your product's ILM configuration.

In addition, the user interface for this master configuration record displays summary information about all the maintenance objects that are configured to use ILM.

Copyright © 2024, Oracle and/or its affiliates.

Confirm the Maintenance Objects to Enable

In viewing the list of maintenance objects that support ILM in the ILM master configuration page, your implementation may choose to enable ILM for only a subset of the supported maintenance objects. For example, some of the maintenance objects may not be relevant for your implementation. Or perhaps, the functionality provided by the maintenance object is used, but your implementation does not expect a high volume of data.

For each maintenance object that your implementation has confirmed for ILM, the following steps should be taken:

- Determine if the maintenance object should have a different number of default retention days than the system wide value defined on the master configuration. If so, use the MO option **ILM Retention Period in Days** to enter an override option for this maintenance object.
- Review the functionality of the ILM Eligibility algorithm provided by the product for this maintenance object. Each algorithm may support additional configuration based on business needs. If your organization has special business rules that aren't satisfied by the algorithm provided by the product, a custom algorithm may be provided to override the base algorithm.

For each maintenance object that your implementation does not want to enable for ILM, inactivate the eligibility algorithm. This will ensure that the ILM Crawler Initiator background process does not submit the crawler batch job for the maintenance object in question.

- Go to the Maintenance Object - Algorithm tab for each maintenance object and take note of the **ILM Eligibility** algorithm code.
- Go to the Maintenance Object - Option tab for the same maintenance object and add an option with an option type of **Inactivate Algorithm** and the value set to the ILM eligibility algorithm noted in the previous step.

Database Administrator Tasks

In order to implement ILM for each of the maintenance objects determined above, your database administrator must perform several steps in the database for the tables related to each MO. The following points are a summary of those steps. More detail can be found in the Information Lifecycle Management section of your product's *Database Administration Guide*.

- **Initializing ILM Date**: Your existing tables that are enabled for ILM may not have the ILM Date and ILM Archive switch initialized on all existing records. When choosing to enable ILM, a first step is to initialize this data based on recommendations provided in the *Database Administration Guide*.
- **Referential Integrity**: The recommended partitioning strategy for child tables in a maintenance object is referential partitioning. In order to implement this, database referential integrity features must be enabled.
- **Partitioning**: This provides a way in which the data can segregate into multiple table partitions and will help in better management of the lifecycle of the data.

Schedule the ILM Crawler Initiator

The final step of enabling the system for ILM is to schedule the ILM crawler initiator **F1-ILMIN** regularly based on your implementation’s need. It is recommended to only schedule this batch process once all the required database activities are complete.

Ongoing ILM Tasks

For an environment where ILM is enabled, besides the periodic execution of the ILM crawler batch processes to review and mark records, your database administrator has ongoing tasks.

The DBA reviews and maintains partitions and identifies partitions that may warrant some type of archiving step. The Information Lifecycle Management section of your product's *Database Administration Guide* provides more information for your DBA.

Copyright © 2024, Oracle and/or its affiliates.

Archived Foreign Keys

If your DBA chooses to archive a partition, there may be records in the system that refer to one of the archived records as a foreign key.

When a user attempts to view a record using a hyperlink or drill down mechanism, if the implementation of the navigation uses FK Reference functionality, the system will first check if the record exists. If not, it will display a message to the user indicating that the record has been archived.

Copyright © 2024, Oracle and/or its affiliates.

Chapter 5

Configuration Tools

Topics:

- Business Objects
- Data Areas
- Business Services
- User Interface (UI) Maps
- Process Flows
- Action Method
- Algorithms
- Script Options
- Maintaining Managed Content
- Defining Entity Tags
- Advanced Schema Topics
- Business Event Log
- Miscellaneous Topics
- Extensions Dashboard

This section describes tools to facilitate detailed business configuration. The configuration tools allow you to extend both the front-end user interface as well as create and define specialized back-end services.

Business Objects

A maintenance object defines the physical tables that are used to capture all the possible details for an entity in the system. A business object is tool provided to further define business rules for a maintenance object.

This section provides an overview of business objects and describes how to maintain them.

Understanding Business Objects

The topics in this section describe background topics relevant to business objects.

What Is A Business Object?

A business object (BO) is a powerful tool used throughout the system. For many maintenance objects, a BO is a key attribute of the record used to define the data it captures, its user interface behavior and its business rules. Some business objects support the definition of a lifecycle, capturing different states that a record may go through, allowing for different business rules to be executed along the way. This type of business object is considered the “identifying” or “governing” business object. We will see later that other types of BOs exist that are different from the “identifying” business object.

The use of business objects allows for extensibility and customization of product delivered maintenance objects. There are many options to adjust the behavior of base delivered business objects. In addition, implementations may introduce their own business objects if the base product delivered objects do not meet their business needs.

Note: Not all maintenance objects in the product support business objects as a “identifying” or “governing” tool. This is the standard going forward for new maintenance objects. However, there are some maintenance objects created before this became a standard.

A Business Object Has a Schema

A business object has elements. The elements are a logical view of fields and columns in one of the maintenance object’s tables. The structure of a business object is defined using an XML schema. The main purpose of the schema is to identify all the elements from the maintenance object that are included in the business object and map them to the corresponding maintenance object fields. Every element in the BO schema must be stored somewhere in the maintenance object. The BO may not define elements that are derived.

When defining elements for the primary table or the language table (for an administrative object) there is no need to define the name of the physical table in the schema. The system infers this information. The following is a snippet of a schema:

```
<schema>
    <migrationPlan mapField="MIGR_PLAN_CD" suppress="true" -
isPrimeKey="true"/>
    <bo mapField="BUS_OBJ_CD" fkRef="F1-BUSOB"/>
    <customizationOwner mapField="OWNER_FLG" suppress="input"/>
    <version mapField="VERSION" suppress="true"/>
    <description mapField="DESCR"/>
    <longDescription mapField="DESCRLONG"/>
  …
```

Many maintenance objects have child table collections (e.g., a collection of names for a person, or a collection of persons on an account). Depending on the requirements, the business object may define the full collection such that the user will maintain the information in a grid. However, the schema also supports “flattening” records in a child table so that they can be treated as if they were singular elements. The following are examples of each:

Copyright © 2024, Oracle and/or its affiliates.

Example of a child table. This is a snippet of the Instructions collection on the migration plan business object. You can see that the list attribute defines the child table and all elements within it map to the appropriate column in that table.

```
   <migrationPlanInstruction type="list" mapChild="F1_MIGR_PLAN_INSTR">
         <migrationPlan mapField="MIGR_PLAN_CD" suppress="true"/>
         <sequence mapField="PLAN_INSTR_SEQ" suppress="true"/>
         <instructionSequence mapField="INSTR_SEQ"/>
         <instructionType mapField="INSTR_TYPE_FLG"/>
         <parentInstructionSequence mapField="PARENT_INSTR_SEQ"/>
         <businessObject mapField="BUS_OBJ_CD" fkRef="F1-BOMO"/>
...
```

Example of a simple “flattened” field. The business object for Status Reason includes an element called Usage, which maps to a pre-defined characteristic of type **F1–SRUSG**. The “row” defines which child table is being flattened and the attributes of the row in that child that uniquely identify it.

```
    <usage mdField="STATUS_RSN_USAGE" mapField="CHAR_VAL"> -
        <row mapChild="F1_BUS_OBJ_STATUS_RSN_CHAR"> -
            <CHAR_TYPE_CD is="F1-SRUSG"/>
            <SEQ_NUM is="1"/> -
        </row>
    </usage>
```

Example of a “flattened row”. This business object for Account includes a single row for the Person collection where only the “financially responsible, main” customer is defined. The “accountPerson” attribute defines one field from that row (the Person Id) and includes the ‘flattening’ criteria in the “row” information. In addition, a second field from that same row (“accountRelType”) is defined. Instead of having to repeat the flattening criteria, the “rowRef” attribute identifies the element that includes the flattening.

```
    <accountPerson mapField="PER_ID">
        <row mapChild="CI_ACCT_PER">
            <MAIN_CUST_SW is="true"/>
            <FIN_RESP_SW default="true"/>
         </row>
     </accountPerson>
     <accountRelType mapField="ACCT_REL_TYPE_CD" rowRef="accountPerson" -
dataType="string"/>
```

Example of a “flattened list”. The business object for Tax Bill Type includes an list of valid algorithms for “bill completion”. The Sequence and the Algorithm are presented in a list. The list element identifies the child table and the ‘rowFilter’ identifies the information about the list that is common.

```
       <taxBillCompletion type="list" mapChild="C1_TAX_BILL_TYPE_ALG">
         <rowFilter suppress="true" private="true">
             <TAX_BILL_TYPE_SEVT_FLG is="C1BC"/>
         </rowFilter>
         <sequence mapField="SEQ_NUM"/>
         <algorithm mapField="ALG_CD" fkRef="F1-ALG"/>
     </taxBillCompletion>
```

In addition, many maintenance objects support an XML structure field within the entity. These fields may be of data type CLOB or XML. One or more business object elements may be mapped to the MO's XML structure field. These elements may be defined in different logical places in the business object schema based on what makes sense for the business rules. When updating the MO, the system builds a type of XML document that includes all the elements mapped to the XML structure and stores it in one column. The following is an example of elements mapped to an XML column:

```
        <filePath mdField="F1_FILE_PATH" mapXML="MST_CONFIG_DATA" -
required="true"/>
```

Copyright © 2024, Oracle and/or its affiliates.

```
        <characterEncoding mdField="F1_CHAR_ENCODING" -
mapXML="MST_CONFIG_DATA"/>
```

Note: If the MO's XML structure field is of the data type XML, the database will allow searching for records based on that data, assuming appropriate indexes are defined. If the MO's XML structure field is of the data type CLOB, indexing or joining to elements in this column via an SQL statement is not typically supported. Note that most MOs are currently using the CLOB data type for the XML structure column, if provided.

Some business objects may have child tables that allow data to be stored in an XML structure field. The schema language supports defining elements from those fields in your schema as well.

Besides including information about the physical "mapping" of the element to its appropriate table / field location in the maintenance object, the schema supports additional syntax to provide the ability to define basic validation and data manipulation rules, including:

- Identifying the primary key of the record or the primary key of the a row in a list.
- Identifying which elements are required when adding or changing a record.
- Default values when no data is supplied on an Add.
- For elements that are lookup values, the lookup may be specified to validate that the value of the element is a valid lookup value.
- For elements that are foreign keys to another table, the FK Reference may be specified to validate the data.

The system will check the validity of the data based on the schema definition obviating the need for any special algorithm to check this validation.

In addition, the schema language may include some attributes that are used to auto-render the view of the record on the user interface, such as the **suppress** attribute. Refer to BO Defines its User Interface for more information.

Note: Refer to Schema Syntax for the complete list of the XML nodes and attributes available to you when you construct a schema.

A business object's schema may include a subset of the fields and tables defined in the maintenance object. There are two reasons for this:

- The fields or tables may not be applicable to the type of record the business object is governing. For example, a field that is specific to gas may not be included on a Device business object that is specific to electric meters.
- The information is not maintained through the business object, but rather maintained separately. For example, many BO based maintenance objects include a Log table. The records in the log table are typically not included the BO because they are viewed and maintained separately from the business object.

A Business Object May Define Business Rules

A business object may define business rules that govern the behavior of entities of this type. The following table describes supported system events that can be associated with such rules.

System Event	Optional / Required	Description
Audit	Optional	Algorithms of this type may be used to audit certain changes made to business object instances. The system hands over to the algorithms a summary of all the elements that were changed throughout a specific call to update an object. For each element its original value before the change as well as its new value are provided. It is the responsibility of such algorithms to log the changes if and where appropriate, for example as a log entry in the maintenance object's log table or an entry in the business event log.

Copyright © 2024, Oracle and/or its affiliates.

System Event	Optional / Required	Description
		These plug-ins are executed after all the post-processing rules are executed. The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to Business Object inheritance for more information. By default all elements of the business object are subject to auditing. You can however mark certain elements to be excluded from the auditing process using the **noAudit** schema attribute. Marking an element as not auditable will prevent it from ever appearing as a changed element in the business object's audit plug-in spot. In addition, if the only elements that changed in a BO are ones marked to not audit, the audit algorithm is not even called. Refer to Schema Syntax for more information on this attribute.
Information	Optional	We use the term "Business Object Information" to describe the basic information that appears throughout the system to describe an entity defined by the business object. The data that appears in this information description is constructed using this algorithm. The system invokes a single algorithm of this type. If more than one algorithm is plugged-in the system invokes the one with the greatest sequence number found on the business object closest to the current business object in the inheritance hierarchy. Refer to Business Object inheritance for more information.
Post-Processing	Optional	Algorithms of this type may be used to perform additional business logic after a business object instance has been processed. For example, such an algorithm may be used to create a To Do Entry or add a log record as part of the business object logical transaction. These plug-ins are executed after all the validation rules are executed. The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to Business Object inheritance for more information.
Pre-Processing	Optional	Algorithms of this type further populates a request to maintain a business object instance right before it is processed. Such algorithm may be used to "massage" a business object's elements prior to any maintenance object processing. For example, although simple element-level defaulting is supported by schema attributes. you may use this type of algorithm to default element values that are more sophisticated. The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to Business Object inheritance for more information.

Copyright © 2024, Oracle and/or its affiliates.

System Event	Optional / Required	Description
Validation	Optional	Algorithms of this type may be used to validate a business object instance when added, updated or deleted. Simple element-level validation is supported by schema attributes. Note that element-level validation is executed before any maintenance object processing. For more sophisticated rules you create validation algorithms and associate them with your business object. BO validation algorithms are only executed after "core validation" held in the MO is passed. The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to Business Object inheritance for more information.

The system applies business object rules (schema based and algorithms) whenever a business object instance is added, changed or deleted. This is only possible when the call is made via the maintenance object service. For example, when made via business object interaction ("invoke BO"), the MO's maintenance page or inbound web services that reference the BO. In addition the system must be able to determine the identifying business object associated with the actual object being processed. If the business object cannot be determined for a maintenance object instance business object rules are not applied.

Note:

Pre-Processing is special. The pre-processing algorithm plug-in spot is unique in that it only applies during a BO interaction. It is executed prior to any maintenance object processing. It means that when performing add, change or delete via the maintenance object service, the pre-processing plug-in is not executed.

CAUTION: Direct entity updates bypass business rules! As mentioned above, it is the maintenance object service layer that applies business object rules. Processes that directly update entities not via the maintenance object service bypass any business object rules you may have configured.

Fastpath: Refer to BO Algorithm Execution Order for a summary of when these algorithms are executed with respect to lifecycle algorithms.

The plug-in spots described above are available for all business objects and they are executed by the system when processing adds or updates to the business object. It is possible for a specific maintenance object to define a special plug-in spot for business objects of that MO. When this happens, the maintenance object identifies the special algorithm entity lookup value as an MO option: **Valid BO System Event**, causing the BO Algorithm collection to include that system event in its list.

A Business Object Defines its User Interface

One of the responsibilities of an identifying business object is to define its user interface rules for viewing and maintenance of its record. The standard implementation for maintaining a business object is that a maintenance portal is used to display a record. This portal includes a "map" zone that displays the information about the business object. To add or make changes to a record, the user clicks a button that launches a maintenance BPA script which displays a maintenance "map".

The display and maintenance "maps" are driven by the business object. The BO may define a full UI map where all the information is displayed based on the map's HTML. Note that for a child BO, the maps may be inherited by a parent BO (or any BO "up the chain").

The standard going forward is to use schema definition and UI Hints to define user interface behavior so that a full UI map is not needed but rather the HTML is derived. The schema language includes some basic display attributes such as **label** and **suppress**. UI hints provide many additional tags and elements that allow dynamic generation of formatted UI

Copyright © 2024, Oracle and/or its affiliates.

Maps. For more complex behavior in the user interface, for example where javascript is needed, UI map fragments may be defined within the schema via UI hints. In this way only complex UI behavior warrants small snippets of javascript and HTML. However the rendering of standard fields can be dynamically rendered. UI map fragments also allow for derived fields to be included in the user interface.

A business object schema may include data areas for segments of its schema definition to allow for reusable components. In this case the data area would also include schema attributes and UI hints for the elements that it is including.

Note: Refer to UI Hint Syntax for detailed information about the supported syntax.

As mentioned in Business Object Inheritance, schemas are not inherited on a child business object. As such, when using UI hints for automatic UI rendering, the child BO must define the full schema with all the definitions. A good business object hierarchy will be designed for reuse meaning that the child BO will include the parent BO schema or alternatively, the BO schemas will include reusable data areas.

A Business Object Defines Options

A business object may be associated with configuration options that control specific features. For example, the scripts used to maintain and display a business object are defined as options. Refer to the detailed description of each option type for more information on the functionality it controls.

Note: You can add new options types. Your implementation may want to add additional option types. For example, your implementation may have plug-in driven logic that would benefit from a new option. To do that, add your new values to the customizable lookup field **BUS_OBJ_OPT_FLG**. If you add a new option type for a business option, you must update its maintenance object to declare this new option type. Otherwise, it won't appear on the option type dropdown. You do that by referencing the new option type as a **Valid BO Option Type** maintenance object option.

Some options may be defined for a specific lifecycle status. For example, marking a status specific base algorithm as inactive is defined as an option on that staus definition record.

Note: You can add new options types. Your implementation may want to add additional option types. For example, your implementation may have plug-in driven logic that would benefit from a new option. To do that, add your new values to the customizable lookup field **BO_OPT_FLG**. If you add a new option type for a status, you must update the business object's maintenance object to declare this new option type. Otherwise, it won't appear on the option type dropdown. You do that by referencing the new option type as a **Valid BO Status Option Type** maintenance object option.

Invoking A Business Object

We have talked about defining a business object. This section describes how business objects are used throughout the system to view, add and update records.

- Various parameters for zones that are used to display data in the system include support for retrieving data by referencing a business object. The zone code will "invoke" the BO, meaning that the record will be retrieved using the referenced BO.
- The system's scripting language includes a step type to "invoke BO". This allows for BPA scripts, service scripts and plug-in scripts to retrieve information and add or update records using BO interaction.
- Inbound web services may reference a business object in its operations collection. This allows external systems to add or update records in our product via web service interaction.

Often when configuring a zone or writing a script, the BO to use in the "invoke BO" statement should be the identifying BO of the record. As such, often the script will include steps prior to the "invoke BO" step to "determine the identifying BO of the record" and once the identifying BO is found, the script step will invoke that BO. Note that zones and inbound web services reference a BO directly. In each case, if the BO to use should be dynamic, then the zone / inbound web service should reference a service script that can perform the steps to identify the BO and then invoke that BO.

It should be noted however that the BO used in an "invoke BO" statement (or referenced in an inbound web service) **does not have to match** the identifying BO for the record. Here are some examples of where this may be true:

Copyright © 2024, Oracle and/or its affiliates.

- A script may only require a subset of elements for a record and not the entire record. In this case, it is better for performance purposes to define a special BO (sometimes called a "lite" BO or a "mini" BO) that only defines the needed elements. When the system retrieves the data, it will only access the tables that are included in the BO's schema definition. In addition, if there are no elements that map to an XML structure field, the system will skip any parsing of that column. Similarly, if a script is **updating** a subset of elements on a record, it may be beneficial to use a "mini" BO to do the updates.

  Note: Please note the following with respect to using a mini BO. This BO is only used for its schema. This type of BO would not define algorithms or a lifecycle. Because the BO is special, it often should not be able to be used as any record's identifying BO. To control that, these BOs are often configured to not allow new instances. Refer to Determine the Identifying BO for more information.

- The maintenance object to be added or updated in a script may not support business objects as "identifying BOs". For example, Batch Control maintenance object does not have an identifying BO. However, scripts may still wish to retrieve data (or make updates) to these types of records. An easy way to achieve that goal is to define a business object and use "invoke BO" to access the data.

  Note: Not all maintenance objects support being maintained through a business object interaction. This is true in a small number of older objects where the underlying maintenance service includes additional functionality besides simply updating the database tables. These maintenance objects are identified via the MO option **BO Maintenance**, set to **N**.

- Some functionality may be trying to add or update records for a maintenance object in a 'physical' manner and do not want or need to use the object's identifying BO. Or the MO may not have an identifying BO. For example, revision control takes a snapshot of a record for audit purpose and to be able to restore a previous version. In this case, the system wants to capture a full "physical" view of the record. To do this, a special "physical" BO may be created that includes all (or most of) the columns and the child tables.

  Note: Like the mini BO, the physical BO would not define algorithms or a lifecycle and should not be able to be used as any record's identifying BO. To control that, these BOs are often configured to not allow new instances. Refer to Determine the Identifying BO for more information.

Note: To reiterate, the BO referenced in the "invoke BO" statement or referenced in an inbound web service does not have to match the identifying BO and does not have to be configured to "allow new instances".

Determine the Identifying BO

As mentioned in other topics, the identifying BO is the business object that governs the business rules for a record. This is the business object that the record will be validated against when any additions or changes are made to the record as long as updates are made via the maintenance service. This includes using "invoke BO" for add or update, using inbound web service interaction and for access to the maintenance page service (via an old style fixed page or via a business service).

How does the system determine the identifying BO? An algorithm plugged into the maintenance object (the **Determine BO** plug-in spot) is responsible for this. If the maintenance object is not configured with an algorithm for this plug-in spot, or no BO is found by the algorithm, no BO business rules are applied.

Most maintenance objects in the system capture the record's identifying BO directly on the record. However, it is possible to define the identifying BO somewhere else. For example, there may be some maintenance objects that are master or transaction objects with an associated "type" object where the identifying BO is defined on its "type" object. Note that the standard Determine BO algorithm plugged into most maintenance objects (**F1-STD-DTMBO - Determine Standard Business Object**) checks for these two conditions.

There may also be cases where a single identifying BO is used for all BOs for a given MO. This may be an option used for some older maintenance object created prior to the business object functionality when implementations wish to introduce custom business rules that are common for all records of that MO. The product provides a base algorithm type (**F1–MOBO - Determine Specific Business Object**) that captures the BO as a parameter.

Copyright © 2024, Oracle and/or its affiliates.

Base Business Objects

For each maintenance object (MO) that supports an "identifying" business object, the type of business object provided by the product depends on the functionality and expected use by implementations. The following are some common patterns.

- There are MOs where the product provides base BOs that implementations may use if applicable for their business rules. In addition, it is expected that implementation will define custom BOs to support their business needs. Good examples of this type of MO are any of the various "rule" MOs. For example, calculation rule in Oracle Utilities Customer Care and Billing or the usage rule in Oracle Utilities Meter Data Management. The product provides business objects for common rules but each implementation could have special rules that they need to implement and will need to create custom business objects.
- There are MOs where the product provides base BOs that supply common behavior for an object. Implementations may find that supplied the business objects match their business requirements and use the BOs as is. It is expected, however that for many implementations, their business rules will require additional elements to be captured or have special rules to apply. In this case the base business objects may be extended. This scenario may apply to 'master' data objects in various products such as the Device or Meter.
- There are MOs where the product may deliver a base BO that is not expected to satisfy most implementations because different jurisdictions or different implementations will typically have their own rules. In this case the base delivered BO can be used as a template or starting point for custom defined BOs. Some examples of this are Rebate Claim in Oracle Utilities Customer Care and Billing.
- There are MOs where the expectation is that every implementation will have different requirements for the type of data to capture and the product will not supply base BOs that can be used as the "identifying" BO. However, it may supply a "parent" BO that defines the lifecycle and many of the business rules that it expects all records to follow. In these scenarios, the implementations will create "child" BOs that will serve as the "identifying" BOs and refer to the base "parent" BO for many of its rules through inheritance. An example of this is the Activity in Oracle Utilities Mobile Workforce Management.
- There are some scenarios where the base product provides business objects and the expectation is that implementations will use the business objects as delivered with little or no customization. This is a case where the system used business objects to implement product functionality, not because there is an expectation that the implementers will extend the functionality, but because the business object model is the favored development tool even for the product. The objects delivered for Content Migration Assistant are an example.

Note: Not all maintenance objects in the product support business objects as a "identifying" or "governing" tool. This is the standard going forward for new maintenance objects. However, there are some maintenance objects created before this became a standard.

For all maintenance objects, the base product may provide additional BOs that are not meant to be "identifying" BOs, but instead are provided to support functionality to interact with the MO using the BO as a tool as described in Invoking a BO.

- One or more "mini" or "lite" BOs may be supplied for a maintenance object. This may be found when the product has functionality to retrieve a subset of elements for the maintenance object via scripting or via a user interface.
- A "physical" BO may be supplied. This a BO that typically includes all tables and all fields of the maintenance object in there "physical" form. In other words, there is no "flattening" of child tables and any XML structure fields are defined as a single field. Physical BOs are used in system processing where the full record needs to be captured as is. Some functionality that uses a physical BO includes bundling, revision control and the pre-compare algorithm for CMA to adjust data prior to comparing. If there are any maintenance objects that do not have a base supplied physical BO, refer to Creating a Physical Business Object for steps on how to provide one.
- A "bundling add" BO may be supplied. Refer to Recursive Key References for more information as to why this type of BO may be supplied.

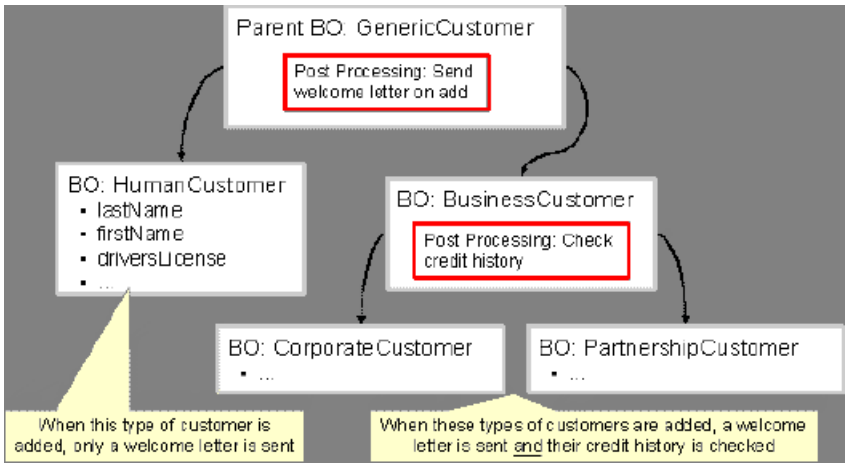
Business Object Inheritance

A business object may inherit business rules from another business object by referencing the latter as its parent. A child business object can also have children, and so on. A parent's rules automatically apply to all of its children (no

Copyright © 2024, Oracle and/or its affiliates.

compilation - it's immediate). A child business object can always introduce rules of its own but never remove or bypass an inherited rule.

The following is an illustration of multiple levels of business object inheritance.



Notice how the "Business Customer" business object extends its parent rules to also enforce a credit history check on all types of customers associated with its child business objects.

Most types of business object system events allows for multiple algorithms to be executed. For example, you can have multiple **Validation** algorithms defined for a business object. For these, the system executes all algorithms at all levels in the inheritance chain starting from the highest-level parent business object moving on to lower levels.

Other types of system events allows for a single algorithm to be executed. For example, you can only have one **Information** algorithm to format the standard description of a business object instance. For these, the system executes the one at the level nearest to the business object currently being processed.

Note: The parent and its children must reference the same maintenance object.

Note: Data structures are not inherited. While you can declare schemas on parent business objects, their children will not inherit them. A good practice is to design child business object schemas to **include** their parent business object's schema.

Note: User interface maps are inherited. When determining if the business object has a UI map to use for UI rendering, the system looks for display and maintenance maps linked to the BO as options. If the identifying BO does not have maps defined, the system follows “up the chain” of inheritance until it finds a map to use. This allows for a child BO to be used to extend business rules of a parent BO but inherit its user interface behavior. Refer to Business Object Defines its User Interface for more information.

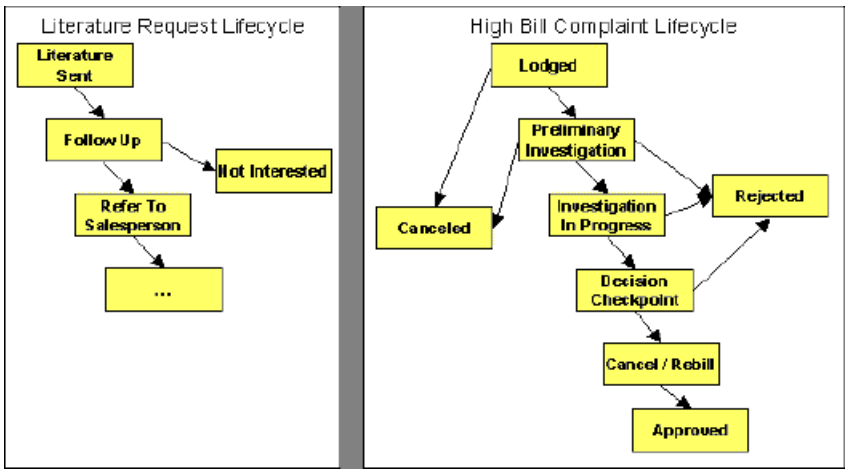
Note: Use Inheritance Wisely. While it is intellectually attractive to abstract behavior into parent BOs to avoid redundant logic and simplify maintenance, before doing this weigh the reuse benefits against the cost in transparency, as it is not easy to maintain a complex hierarchy of business objects.

Each Business Object Can Have A Different Lifecycle

Many maintenance objects have a status column that holds the business entity's current state within its lifecycle. Rules that govern lifecycle state transition (what is its initial state, when can it transition to another state, and so on.) and the behavior associated with each state are referred to as lifecycle rules. Older Maintenance Objects, such as To Do Entry, have predefined lifecycles whose rules are governed by the base-package and cannot be changed. The lifecycle of newer Maintenance Objects exists in business object meta-data and as such considered softly defined. This allows you to have completely different lifecycle rules for business objects belonging to the same maintenance object.

Here are examples of two business objects with different lifecycles thast belong to the same maintenance object.

Copyright © 2024, Oracle and/or its affiliates.



The boxes in the above diagram show the potential valid states a business entity of the above business object can be in. The lines between the boxes indicate the state transition rules. These rules govern the states it can move to while in a given state. For example, the above diagram indicates a high bill complaint that's in the **Lodged** state can be either **Canceled** or moved into the **Preliminary Investigation** state.

When you set up a business object, you define both its valid states and the state transition rules, including the label for the Action to display on a button for users that may initiate a transition. For example, the action label to transition to the 'Approved' state could be configured as 'Approve'.

Note: The status description and the action label of base delivered values may be overridden by your implementation.

Note: A Maintenance Object supports soft lifecycle rules if it is defined with a **Status Field** Maintenance Object option.

The topics that follow describe important lifecycle oriented concepts.

One Initial State and Multiple Final States

When you set up lifecycle states, you must pick one as the initial state. The initial state is the state assigned to new entities associated with the business object. For example, the above high-bill complaint business object defines an initial state of **Lodged**, whereas the literature request one defines an initial state of **Literature Sent**.

You must also define which statuses are considered to be "final". Typically when an entity reaches a "final" state, its lifecycle is considered complete and no further processing is necessary.

Note: Allowing An Entity To Be "Reopened". You can set up your state transition rules to allow a business entity to be "reopened" (i.e., to be moved from a final state to a non-final state). Neither of the above examples allows this, but it is possible if you configure your business object accordingly.

State-Specific Business Rules

For each state in a business object's lifecycle, you can define the following types of business rules.

System Event	Optional / Required	Description
Enter	Optional	Algorithms of this type apply business rules when a business object instance enters a given state.
Exit	Optional	Algorithms of this type apply business rules when a business object instance exits a given state.
Monitor	Optional	Algorithms of this type monitor a business object instance while in a given state. Typically these are used to auto-transition it to another state.

Copyright © 2024, Oracle and/or its affiliates.

The system invokes all algorithms of this type defined on the business object's inheritance hierarchy. Refer to Business Object Inheritance for more information.

Fastpath: Refer to BO Algorithm Execution Order for a summary of when these lifecycle algorithms are executed with respect to BO level algorithms.

Entering A State

You can define algorithms that execute before a business entity enters a given state. For example, you could develop an algorithm that requires a cancellation reason before an entity is allowed to enter the **Canceled** state.

You can also incorporate state auto-transitioning logic within this type of algorithms. Refer to auto-transition for more information.

Also note that when a record is processed by the monitor batch program, by default the BO Post Processing, BO Audit and MO Audit algorithms are not executed. However, it is possible for an enter algorithm to indicate that the other algorithms should be executed by the batch process by setting the “force post processing” indicator to true.

Exiting A State

You can define algorithms that execute when a business entity exists a given state. For example, you could develop an algorithm that clears out error messages when a given entity exits the **Error** state.

Also note that when a record is processed by the monitor batch program, by default the BO Post Processing, BO Audit and MO Audit algorithms are not executed. However, it is possible for an exit algorithm to indicate that the other algorithms should be executed by the batch process by setting the “force post processing” indicator to true.

Monitor Rules

You can define algorithms to monitor a business entity while it is in a given state. This type of logic is typically used to check if the conditions necessary to transition the entity to another state exist (and, if so, transition it). For example, transition an entity to the **Canceled** state if it's been in the **Error** state too long. Another common use is to perform ancillary work while an entity is in a given state. For example, update statistics held on the object while it's in the **Active** state.

Monitor algorithms are invoked when a business entity first enters a state and periodically after that in batch. You have the option to defer the monitoring of a specific state until a specific monitoring batch job runs. You do so by associating the state with a specific monitoring process. In this case the system will only execute the monitoring rules of this state when that specific batch process runs. This is useful when processing one type of record typically creates another type of record. You may want the processing of the second set of records to be deferred to a later time.

A monitor algorithm can carry out any business logic. In addition it can optionally tell the system to do either of the following:

- Stop monitoring and transition to another state. The system will not call any further monitoring algorithm plugged in on the state and attempt to transition the entity to the requested new state.
- Stop monitoring. Same as above except that no transition takes place. You may want to use this option to prevent transitions while some condition is true.

If none of the above is requested the system keeps executing subsequent monitoring algorithms.

Also note that when a record is processed by the monitor batch program, by default the BO Post Processing, BO Audit and MO Audit algorithms are not executed. However, it is possible for a monitor algorithm to indicate that the other algorithms should be executed by the batch process by setting the “force post processing” indicator to true.

Fastpath: Refer to Business Object - Lifecycle for more information on how to set up state-specific algorithms.

Inheriting Lifecycle

If a business object references a parent business object, it always inherits its lifecycle from the highest-level business object in the hierarchy. In other words, only the highest-level parent business object can define the lifecycle and the valid state transitions for each state. Child business objects, in all levels, may still extend the business rules for a given state by introducing their own state-specific algorithms.

Copyright © 2024, Oracle and/or its affiliates.

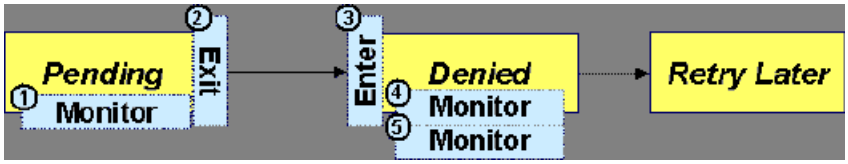
The system executes all state-specific algorithms at all levels in the inheritance chain starting from the highest-level parent business object moving on to lower levels.

Auto-Transition

In a single transition from one state to another, the system first executes the **Exit** algorithms of the current state, transitions the entity to the new state, executes the **Enter** algorithms of the new state followed by its **Monitor** algorithms. At this point if a **Monitor** algorithm determines that the entity should be further automatically transitioned to another state the remaining monitoring algorithm defined for the current state are not executed and the system initiates yet another transition cycle.

Notice that an **Enter** algorithm can also tell the system to automatically transition the entity to another state. In this case the remaining **Enter** algorithm as well as all **Monitor** algorithms defined for the current state are not executed.

The following illustration provides an example of an auto-transition chain of events.



In this example a business entity is in a Pending state. While in that state a **Monitor** algorithm determines to auto-transition it to the Denied state. At this point the following takes place:

- No further **Monitor** algorithms of the Pending state are executed
- Pending state **Exit** algorithms are executed
- The system transitions the entity to the Denied state
- Denied state **Enter** algorithms are executed. No further auto-transition is requested.
- Denied state **Monitor** algorithms are executed. No further auto-transition is requested.

Keeping An Entity In Its Last Successful State

By default, any error encountered while transitioning a business entity from one state to another rolls back all changes leaving the entity in its original state.

When applicable, the Maintenance Object can be configured to always keep an entity in its last successful state rather than rolling all the way back to the original state. This practice is often referred to as "taking save-points". In case of an error, the entity is rolled back to the last successfully entered state and the error is logged on the maintenance object's log. Notice that with this approach no error is returned to the calling process, the error is just logged.

The logic to properly log the error is in a **Transition Error** Maintenance Object plug-in. The system considers a maintenance object to practice "save-points" when such an algorithm is plugged into it.

Even if the maintenance object practices "save-points", in case of an error the system will not keep an entity in the last successfully entered state if that state is either marked as transitory or one of its **Enter** algorithms has determined that the entity should proceed to a next state. The system will roll back to the first previous state that does not match these conditions.

Monitoring Batch Processes

A monitor batch process may be used to transition a business object into its next state by executing the monitor algorithms associated with the current state of the entity. The use cases for performing the monitor logic in batch are as follows:

- The record may be waiting for something else to occur before transitioning. The monitor algorithm may be coded to determine if the condition is satisfied and initiate the transition then. For example perhaps when entering a state, a field activity is generated and the record should exit the state when the field activity is complete. The monitor algorithm can check the status of the field activity.
- Perhaps a record is added or updated manually and the next step in the BO lifecycle includes a large amount of processing such that the logic should occur in batch. In this case the BO status is configured with an explicit reference to a batch control (referred to as "deferred"), which indicates to the system that the monitor algorithms should not

Copyright © 2024, Oracle and/or its affiliates.

be performed automatically (but should be deferred to batch). Later when the batch process runs, it selects all the records to process to progress the records.

> **Note:** When a status includes a deferred batch control, it may also be configured to allow a user to manually transition the record to the next state rather than waiting for batch. When a user manually transitions a record that includes monitor algorithms, those algorithms are not executed.

- Perhaps a record is added or updated in batch, but a subsequent step in the overall lifecycle should be processed later. This may be accomplished by ensuring that the batch control linked to the state to process later does not match the batch control that added or updated the record.
- Monitor processes may also be used to periodically perform some logic related to the record without actually transitioning the record.

Note that only the parent business object may refer to a deferred monitor batch process. However, any business object in the "inheritance" chain may be configured with monitor algorithms, which will all be executed.

The base package provides a periodic monitoring batch process for each maintenance object that supports a configurable BO lifecycle. The process periodically executes the monitoring algorithms associated with the current state of an entity, excluding states explicitly referencing a deferred monitoring batch process that is for a different batch control.

A deferred monitoring process selects records whose current state references this particular batch control as their monitor process. Deferred monitor process is only needed when an object has different use cases for monitoring the same type of records with different schedules. In this case only one periodic monitor batch should be configured. Other monitors should be configured to restrict by batch control so that there is no overlap in processing.

> **Note: MO option configuration**. The maintenance object includes options to indicate the batch controls delivered for periodic and deferred monitor batch controls.

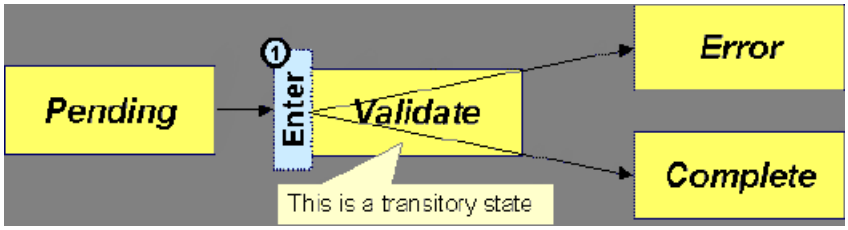
Your business rules will dictate the execution frequency of each monitoring process and the order in which they should be scheduled. Refer to Monitor Background Processes in the background process chapter for more information about the parameters supported for this type of batch process.

> **Note: Updates to the business object.** When the monitor algorithms indicate that the business object should transition, the monitor batch processes are responsible for ensuring the business object is transitioned appropriately and that the appropriate exit, enter and monitor algorithms are executed. Please note that the business object is not updated using a call to the maintenance object service and therefore by default the business rules plugged in to the business object are not executed. However, it is possible for an Enter algorithm, Exit algorithm or Monitor algorithm to indicate that the other algorithms should be executed by the batch process. If the "force post processing" indicator is set to true, then the batch process invokes the BO Post Processing, BO Audit and MO Audit algorithms.

Transitory States

You can define a state as **Transitory** if you do not wish the business entity to ever exist in that particular state.

The following illustrates a lifecycle with a transitory Validate state.



In this example, the business entity is saved still not validated in the Pending state. At some point, the user is ready to submit the entity for validation and transitions it into a transitory state Validate whose **Enter** rules contain the validation logic. The responsibility of the transitory state's **Enter** algorithms is to decide if the entity is valid or in error and then transitions it into the appropriate final state. In this scenario, you may not ever want the business entity to exist in the Validate state.

Copyright © 2024, Oracle and/or its affiliates.

Let's also assume that the maintenance object in this example is practicing "save-points" and requires the entity to be kept in its last successful state. If an error were to occur during the transition from **Validate** to the next state, the system would roll back the entity back to Pending, and not Validate even though the entity has successfully entered the Validate state. Refer to the Auto Transition section for more information.

State Transitions Are Audited

Most Maintenance Objects that support soft lifecycle definition also have a log to hold significant events throughout a business entity's lifecycle. For example, log entries are created to record:

- When the business entity is created (and who created it)
- When its status changes (and who changed it)
- If a transition error occurred (and the error message)
- References to other objects created throughout the entity's lifecycle. For example, if a To Do entry is created as part of processing the entity, the To Do Entry is referenced in the log.
- Manual entries added by a user (think of these as "diary" entries)

When a business entity is first created and when it transitions into a new state the system calls **Transition** algorithm(s) plugged in on the Maintenance Object to record these events. If the maintenance object supports a log these events can be captures as log entries.

Note: Most base package maintenance objects supporting a log may already provide state transition logging as part of their core logic. In this case you only need to provide a **Transition** plug-in if you wish to override base logging logic with your own.

Required Elements Before Entering A State

You can define additional elements that are required before a business entity can enter a given state. For example, let's assume that a Cancel Reason must be defined before an object can enter the *Canceled* state. You do this by indicating that element as a **Required Element** state-specific option on the appropriate state on the business object.

Capturing a Reason for Entering a State

Some business objects support configuring certain states to allow or require a status reason when an object enters the state. The product provides a centralized status reason table that may be used to define the valid BO status reasons for various business objects and various states. The status reasons are defined using the Status Reason portal.

The following sections provide additional information about the BO status reason functionality.

Maintenance Object Must Support Status Reason

In order for a business object to use the centralized status reason table to define reasons, the maintenance object must first support the status reason. MOs that support status reason have the following characteristics:

- The primary table includes a column for Status Reason. This represents the status reason for the record’s current status, if applicable.
- The log table includes a column for Status Reason. The standard logic for capturing a log record when entering a state also captures the status reason, if applicable. This allows a user to review the history of the changes in status and the status reason captured for a previous state transition, if applicable.
- The maintenance object option collection includes an option that defines the Status Reason field. This setting is a trigger for business objects of this MO to be able to configure states to allow or require status reason.

Business Object State Indicates if Reasons are Applicable

Once the MO is configured to support status reason, configuration on the business object is required to indicate the states where a reason is applicable. States may be configured to require status reasons, allow status reasons as optional or not allow status reasons. With this configuration, the framework will automatically get the list of valid reasons for a state that allows or requires them and then prompt the user for a status reason when a manual state transition occurs for that state. It also automatically triggers an error if the state requires a status reason and no reason is provided.

Copyright © 2024, Oracle and/or its affiliates.

Note: The status reason configuration on the business object state is customizable. That means that for a product owned business object, an implementation may opt to change the delivered configuration.

Status reasons are defined for the parent (or "lifecycle") business object. All business objects in the hierarchy of the parent business object have the same valid reasons for their states.

The status reason code must be unique for the centralized status reason table. Business object and status are required fields, so it is not possible to share a common reason code (like "Not applicable") across multiple business objects or states. If multiple BOs / states want to support a reason "Not applicable" then each must define a unique record for it. This point should be considered when planning for your status reasons.

Override Description

The description of product owned status reasons may be overridden by your implementation.

Selectable vs. Not Selectable

When defining a status reason, you may indicate whether it's **Selectable** or **Not Selectable**. When a manual transition is performed and a user is prompted for a status reason, only the **Selectable** reasons are presented. The **Not Selectable** reasons may be defined to support transitions that occur via algorithm processing.

Note: The Selectable setting is customizable. That means that if a product provides a base owned status reason for a business object state, an implementation may opt to change whether it is selectable or not. Careful consideration should be made before changing a base delivered status reason from **Not Selectable** to **Selectable** as this may affect base provided algorithm functionality that could be relying on the setting of **Not Selectable**.

Status Reason Business Object

The status reason maintenance object, as with many maintenance objects in the product, references a business object used to define attributes and behavior related to defining status reasons. The framework provides a business object for status reason (**F1–BOStatusReason**). For the business objects that have states that require a status reason (let's call these "transactional BOs"), if there is some special logic required for defining the status reasons, it is possible to define a different status reason BO. In this situation, the override status reason BO to use for capturing status reasons should be defined as a BO option on the transactional BO using the **Status Reason Business Object** option type. If a transactional BO does not define any status reason BO option, then the **F1–BOStatusReason** is used when adding a status reason.

Defining a Usage

The base product status reason BO provides the ability to define a "usage" value. This is useful for algorithms that perform state transitions where a status reason is needed and where the algorithm is usable by more than one business object. In this case, the status reason to use cannot be provided as a parameter because each business object must define its own set of status reasons for each state. The Usage value can be used instead. Each business object can configure the status reason to use in the algorithm and set the appropriate usage value. The algorithm can reference the usage value and retrieve the correct status reason to use based on the record's transactional BO.

The status reason business object provided with the framework product (**F1–BOStatusReason**) supports capturing a usage. The valid usage values are defined in the **Status Reason Usage** characteristic type.

Alternatives for Defining Reasons

There may be business objects in the system that capture reasons that are defined somewhere besides the BO status reason table. For example, some objects may have an explicit administrative table for status reasons. Some objects may use a Lookup or an Extendable Lookup to capture reasons. Refer to the business object description for information about how valid reasons are defined, if applicable.

If a business object supports a reason that is not related to a state transition (such as a creation reason), the BO status reason would not be used. One of the alternate methods for defining a reason, described above, would be used.

Copyright © 2024, Oracle and/or its affiliates.

Opening UI Maps Before Entering A State

You can define a UI Map to capture additional elements before a business entity can enter a given state. You do this by configuring the **State Transition UI Map** state-specific option on the appropriate state on the business object.

There may be circumstances in which the status pre-processing logic can determine the value of the elements, in which case it is not necessary to invoke the map. The system provides a data area (**F1-StateTransitionCommon**) which contains a boolean element to indicate if the state transition map can be skipped. This data area can be included in your pre-processing script to allow the element to be set according to your business logic. The base maintenance script (**F1-MainProc**) references this element to determine whether to open the state transition map.

BO Algorithm Execution Summary

This table highlights the processing steps that occur when adding or changing a record that is governed by a business object.

Invoke BO	
Event	**Comments**
BO Pre-processing algorithms executed	These algorithms are only executed when Invoke BO is used. The business object in the Invoke BO is the one whose rules are executed.

MO Processing	
Event	**Comments**
Determine if status has changed.	The system keeps a note of the new status value but initially proceeds with the old value.
MO Processing.	Standard MO processing, including MO validation is executed.
Determine BO algorithm executed.	The MO level algorithm is executed to determine the identifying BO.
BO Validation algorithms executed.	
State transition rules are performed if the status has changed.	BO Status Exit algorithms for the “old” status executed.
	Status updated to the new value.
	BO Status Enter algorithms for the “new” status executed.
	If no error - MO Transition algorithms are executed.
	Fastpath: Refer to State Transitions are Audited for more information.
	If error and there are “save points” the MO Transition Error algorithms are executed.
	Fastpath: Refer to Keeping An Entity In Its Last Successful State for more information.
	Otherwise, the error is reported.

Copyright © 2024, Oracle and/or its affiliates.

BO Status Monitor algorithms are executed.	If the record transitions again, the prior step (State transition rule step) is repeated for the new transition.
BO Post-processing algorithms are executed.	
BO Audit algorithms are executed.	These algorithms are only executed if the system detects a change in elements that are not marked with “no audit”.

Note: To emphasize, the steps in the MO Processing table are only executed when the maintenance object service is invoked. Any add or update initiated by an “invoke BO” statement will invoke the MO service. This is also true for web service that invoke the business object. The Monitor Batch Process does not invoke the maintenance service. By default the monitor batch process only executes the monitor algorithms and the state transition rules (if the monitor algorithms indicate that a status change should occur). However, it is possible for an Enter algorithm, Exit algorithm or Monitor algorithm to indicate that the other algorithms should be executed by the batch process. If the “force post processing” indicator is set to true, then the batch process invokes the BO Post Processing, BO Audit and MO Audit algorithms.

Note: For records that do not have a status, the state transition rules and the monitor rules are not applicable.

Granting Access To Business Objects

Every business object must reference an application service. When you link a business object to an application service, you are granting all users in the group access to its instances. You can prevent users from transitioning a business object into specific states by correlating each business object status with each application service action (and then don't give the user group rights to the related action).

Fastpath: Refer to The Big Picture Of Application Security for information about granting users access rights to an application service.

The system checks if a user has access rights each time the application is invoked to add, change, delete, read, or transition a business object. However, if a business object invokes another business object, we assume that access was controlled by the initial business object invocation and we do not check access for other business objects that it invokes. In other words, access rights are only checked for the initial business object invoked in a service call.

In order to apply business object security the system must be able to determine the business object associated with the actual object being processed. To do that the Maintenance Object itself has to have a **Determine BO** algorithm plugged in. If this algorithm is not plugged in or it cannot determine the BO on the MO, the system will not invoke any BO rules. If the business object cannot be determined for a maintenance object instance, business object security is not checked. In this case the system checks the user's access rights using standard maintenance object security.

Note: Parent business objects are ignored. If a child business object exists, a user need only have access to the child business object's application service (not to every application service in the business object hierarchy).

Defining Business Objects

This portal is used to view and maintain information about a business object.

You can access the portal from the **Admin Menu** > **System** > **Business Object**. You are brought to a query portal with options for searching for a specific business obejct. Once a business object has been selected you are brought to the maintenance portal to view and maintain the selected record.

Refer to Understanding Business Objects for more information.

CAUTION: Important! If you introduce a new business object, carefully consider its naming convention. Refer to System Data Naming Convention for more information.

Copyright © 2024, Oracle and/or its affiliates.

The following zones may appear as part of the portal's **Main** tab page.

- **Business Object.** This zone provides information about the selected business object.
- **Schema Editor.** This zone describes the business object's schema and supports basic edits in plain text. Refer to the **Schema** tab below for additional schema editor information.

The following zones may appear as part of the portal's **Schema** tab page.

- **Schema Editor.** This zone allows you to edit the business object's schema using the Schema Designer tool. The purpose of the schema is to describe the business object's properties and map them to corresponding maintenance object fields.

> **Fastpath:** Refer to Schema Syntax and UI Hint syntax for a complete list of the XML nodes and attributes available to you when you construct a schema.

> **Tip:** Click the **View Schema** hyperlink to view the business object's expanded schema definition. Doing this opens the schema viewer window.

The following zones may appear as part of the portal's **Algorithms** tab page. Refer to A Business Object May Define Business Rules for more information.

- **Algorithms.** This zone is used to maintain the algorithms associated with the current business object.
- **Hierarchy Algorithms.** This zone displays all algorithms associated with the business object's inheritance hierarchy. The zone is visible when the current business object is part of a hierarchy and such algorithms exist.

The following zones may appear as part of the portal's **Options** tab page. Refer to A Business Object Defines Options for more information.

- **Options.** This zone is used to maintain various configuration options associated with the current business object.
- **Hierarchy Options.** This zone displays all options associated with the business object's inheritance hierarchy. The zone is visible when the current business object is part of a hierarchy and such options exist.

The following zones may appear as part of the portal's **Lifecycle** tab page. This tab is only available when the maintenance object supports lifecycle definition.

- **Lifecycle.** This zone describes and maintains the current business object's lifecycle definition. Broadcast a specific status to review its complete configuration.
- **Status Details.** This zone is used to maintain basic definition information for a selected status within the current business object's lifecycle.
- **Status Algorithms.** This zone is used to maintain algorithms for a selected status within the current business object's lifecycle.
- **Status Options.** This zone is used to maintain various configuration options for a selected status within the current business object's lifecycle.
- **Hierarchy Status Algorithms.** This zone displays all algorithms associated with the business object's inheritance hierarchy for a selected status. The zone is visible when the current business object is part of a hierarchy and such algorithms exist.
- **Hierarchy Status Options.** This zone displays all options associated with the business object's inheritance hierarchy for a selected status. The zone is visible when the current business object is part of a hierarchy and such options exist.
- **Status Reasons.** This zone is used to maintain reasons for a selected status. The zone is visible when the maintenance object supports a reason field and usage of reasons is allowed by the status definition.

The following zones may appear as part of the portal's **Lifecycle View** tab page. This tab is only available when the maintenance object supports lifecycle definition.

- **Lifecycle View.** This zone provides a schematic view of the current business object's lifecycle definition. Broadcast a specific status to review its complete configuration.
- **Status Details.** This zone is used to maintain basic definition information for a selected status within the current business object's lifecycle.
- **Status Algorithms.** This zone is used to maintain algorithms for a selected status within the current business object's lifecycle.

Copyright © 2024, Oracle and/or its affiliates.

- **Status Options.** This zone is used to maintain various configuration options for a selected status within the current business object's lifecycle.

The following zones may appear as part of the portal's **Hierarchy** tab page. This tab is only available when the current business object is part of a hierarchy.

- **Business Object Hierarchy.** This zone lists the inheritance hierarchy of business objects up to and including the current business objects. The zone is visible if the current business object is not the top level business object of the hierarchy.
- **Business Object Tree.** This zone describes the entire hierarchy of the current business object.
- **Hierarchy Algorithms.** This zone lists all algorithms defined for the current business object. The zone is visible if algorithm information exists for the current business object.
- **Hierarchy Lifecycle Algorithms.** This zone lists all algorithms associated with the current business object's inheritance hierarchy. The zone is visible if the current business object is not the top level business object of the hierarchy and algorithm information exits. Refer to Inheriting Lifecycle for more information.
- **Hierarchy Options.** This zone lists all configuration options for the current business object's lifecycle. The zone is visible if options information exists for the current business object.
- **Hierarchy Lifecycle Options.** This zone displays all options associated with the current business object's inheritance hierarchy. The zone is visible if the current business object is not the top level business object of the hierarchy and options information exits. Refer to Inheriting Lifecycle for more information.

The following zones may appear as part of the portal's **References** tab page.

- **Business Object References.** This zone list various configuration entities referencing the current business object or its schema.

Actions

The page supports the following special actions:

- **Generate User Interfact Configuration Artifacts.**When creating a new buisiness object or editing an existing one, you may request to generate the user interface configuration artifacts needed to properly maintain and display the business object on an existing portal.

  For example, if a display script is needed to further prepare the business object's details for presentation then request to generate such script. You can then navigate to the script from the **User Interface Configuration** section and adjust it as needed.
- **Subclass a Business Object.** Use the **Subclass** action to create a new business object that is a child of an existing business object. The action generates the new business object's schema as inherited from the existing business object. Refer to Business Object Inheritance for more information.

  Once created, you may extend and adjust the new business object's schema and rules as needed.
- **Create an Algorithm.** Use this action, on the Algorithm tab or the Lifecycle tab for a specific status, to creates a script based algorithm related to this business object. You are prompted for information regarding the plug-in spot (BO or BO Status) and the system event, the name, description, etc. Once all the information is provided, the system creates an algorithm type, a script, an algorithm and links the algorithm to the business object. You may then navigate to the script to adjust it as needed.
- **Inactivate / Activate Base Algorithms.** You can inactivate algorithms on base Business Objects. Your implementation may want to use a business object provided by the base product, but may want to inactivate one or more algorithms provided by the base business object. To do that, use the "Inactivate" action on the appropriate algorithm entry on the Algorithm tab. Use use the "Activate" action to mark an inactive algorithm as active.

  Note that these actions add and remove the "Inactive Algorithm" businesss option as needed.
- **View Schema.** Click the hyperlink to view the business object's expanded schema definition. Doing this opens the schema viewer window.

Copyright © 2024, Oracle and/or its affiliates.

Advanced BO Tips and Techniques

The topics in this section describe some advanced tips and techniques for configuring business objects.

Creating a Physical Business Object

The physical business object's schema represents the complete physical structure of the maintenance object, and includes elements for all fields in the maintenance object's tables. There are several processes in the system that may use a maintenance object's physical BO.

To create a physical business object for the maintenance object:

1. Navigate to Business Objectand add a business object for the maintenance object. Do not fill in any of the UI oriented configuration.
2. The business object is created with a schema that looks like the physical structure of the maintenance object.
3. Save the physical business object.
4. Navigate to Maintenance Object and search for the maintenance object.
5. On the **Option** tab, add a new option with the type **Physical Business Object**. The value is the name of the physical business you just created.

Managing To Do Entries

The product provides several base algorithm types that may be used to manage To Do entries through status changes for a given record via BO lifecycle plug-ins.

Create To Do Entry

The product supplies a BO status Enter algorithm type **Generic To Do Creation** (**F1-TDCREATE**) that creates a To Do entry based on parameter configuration. Refer to the algorithm type description for more information about how it determines the To Do type or To Do role and how to populate the appropriate message text onto the To Do. This algorithm may be used in conjunction with the Retry Logic (below).

If your implementation has a business rule that requires a To Do entry to be created when entering a given BO status and the logic provided by the algorithm type meets the needs of the business rule, this algorithm type may be used. Create an algorithm for the algorithm type, populate the algorithm parameters according to the business rules and plug the new algorithm into the appropriate business object status as an Enter algorithm.

Retry Logic

The algorithm type **Retry for To Dos** (**F1-TODORETRY**) is supplied for a special use case. It is a BO status monitor plug-in and may be used for a state that is a type of ‘error’ or ‘waiting’ state. It relies on the To Do entry creation logic to set a Retry Frequency. The algorithm transitions to the originating state to retry the logic. The idea is that the condition that caused the record to enter the ‘error’ or ‘waiting’ state may be resolved after some period of time has passed, allowing the record to progress in its lifecycle. Refer to the algorithm type description for more information about its logic.

To use this functionality, create an algorithm for this algorithm type, populate the algorithm parameters according to the business rules and plug the new algorithm into the appropriate business object status as a Monitor algorithm. The state should also have an algorithm for the **Generic To Do Creation** algorithm type plugged in as an Enter algorithm (or something equivalent) that sets the appropriate Retry Frequency.

To Do Completion

It is common that one or more To Do entries associate with a given record should be completed when exiting a state (if it is not already completed). The system supplies the algorithm type **Generic To Do Completion** (**F1-TODOCOMPL**) that may be used for this purpose. Note that the algorithm type functionality is not tied to any To Do creation logic. It may be used for any use case where To Do entries should be completed on exiting a state. Refer to the algorithm

Copyright © 2024, Oracle and/or its affiliates.

type description for more information about its functionality and how to prevent certain To Do entries from being automatically completed.

To use this functionality, create an algorithm for this algorithm type, populate the algorithm parameters according to the business rules and plug the new algorithm into the appropriate business object status as an Exit algorithm.

Submitting a Batch Job

The product provides a base algorithm type that submits a batch job when entering a BO state. This functionality allows for “event driven” batch submission where the event is the lifecycle transition for a certain record.

The algorithm type is **Create Batch Job Submission Entry for Batch Control** (**F1-SCHEDJOB**). The batch control code is a parameter for the algorithm. Refer to the algorithm type description for more information about its logic.

To use this functionality, create an algorithm for this algorithm type, populate the algorithm parameter with the batch control that should be submitted and plug the new algorithm into the appropriate business object status as an Enter algorithm.

Defining Status Reasons

Status Reasons are used to provide more information about why a business object transitioned to a given state. The status reason table provides a centralized place where status reasons can be defined across many different business objects and states.

Note: Refer to Defining Reasons for Entering a State for overview information.

If a business object has one or more states that are configured to capture a status reason, you may configure the valid reasons by navigating to the status reason portal using **Admin** > **System** > **Status Reason**.

The topics in this section describe the base-package zones that appear on the Status Reason portal.

Business Objects with Status Reason List

This zone displays the business objects that have one or more status values that allow status reasons to be defined.

Click the broadcast icon to open other zones that contain more information about the business object’s status reasons.

Status Reasons

The **Status Reasons** zone contains a list of the existing status reasons for the broadcasted business object.

Data Areas

The data area has no business purpose other than to provide a common schema location for re-used schema structures. It exists solely to help eliminate redundant element declaration. For example, if you have multiple schemas that share a common structure, you can set up a stand-alone data area schema for the common elements and then include it in each of the other schemas.

Be aware that a stand-alone data area can hold elements that are mapped to true fields. For example, you might have 50 different types of field activities and all might share a common set of elements to identify where and when the activity should take place. It would be wise to declare the elements that are common for all in a stand-alone data area and then include it in the 50 field activity business objects.

It's strongly recommended that you take advantage of stand-alone data areas to avoid redundant data definition!

CAUTION: Dynamic inclusion! When the system renders a schema, all schemas included within it are expanded real-time. This means that any change you make to a data area will take effect immediately within all schemas it is referenced within.

Copyright © 2024, Oracle and/or its affiliates.

> **Note:**
>
> **Schema Tips**. The data area page includes a special Schema Tips zone that provides a link to launch help topics related to the Advanced Schema Topics help in one click.

Data areas may be included in a business object that does not define a full UI map for display or input. Rather, it is using auto-rendering by defining UI attributes in its schema and via UI hints.

> **Note: View UI Rendering.** A context sensitive "View UI Rendering" zone appears on this page. It may be used for a data area that is part of a business object that is using auto-rendering for the display and input maps. The buttons allow you to view the rendered UI for the segment of the schema that is defined by the data area.

Defining Data Areas

The topics in this section describe how to maintain Data Areas.

Data Area - Main

Use this page to define basic information about a data area. Open this page using **Admin** > **System** > **Data Area**.

Use the **Data Area Query** portal to search for an existing data area. Once a data area is selected, you are brought to the maintenance portal to view and maintain the selected record.

Description of Page

Enter a unique **Data Area** name and **Description**. Use the **Detailed Description** to describe what this data area defines in detail. **Owner** indicates if the data area is owned by the base package or by your implementation (**Customer Modification**).

> **CAUTION:** Important! If you introduce a new data area, carefully consider its naming convention. Refer to System Data Naming Convention for more information.

Click the **View Schema** to view the data area's expanded schema definition. Doing this opens the schema viewer window.

Click the **View XSD** hyperlink to view the business object's expanded schema definition in XSD format.

To extend another data area, reference that data area in the **Extended Data Area** field. By extending a data area you can add additional elements to a base product data area.

Here's an example of an extended data area:

- The product releases with data area A, which contains elements a, b, and c.
- Your implementation creates data area CM-A, which contains element z, and references data area A as the extended data area.
- At run time, everywhere data area A is included it will contain elements a, b, c, and z.

Data Area - Schema

Use this page to maintain a Data Area's schema and to see where the data area is used in the system. Open this page using **Admin** > **System** > **Data Area** and then navigate to the **Schema** tab.

Description of Page

The contents of this section describe the zones that are available on this portal.

The **General Information** zone displays the main attributes of the data area.

The Schema Designer zone allows you to edit the data area's schema. The purpose of the schema is to describe the structure and elements of the data area.

Copyright © 2024, Oracle and/or its affiliates.

Fastpath: Refer to Schema Syntax and UI Hint syntax for a complete list of the XML nodes and attributes available to you when you construct a schema. Also note that the **Schema Tips** zone in the sidebar provides links to launch these help topics directly.

Note: View UI Rendering. A context sensitive "View UI Rendering" zone is associated with this page. The zone is useful for data areas that are to be included in business objects that define the user interface detail using schema attributes and UI Hints. The buttons allow you to view the automatically rendered display and input maps.

The **Schema Usage Tree** zone summarizes all cross-references to this schema. These may be other schemas, scripts, and web services. For each type of referencing entity, the tree displays a summary node showing a total count of referencing items. The summary node appears if at least one referencing item exists. Expand the node to list the referencing items and use their description to navigate to their corresponding pages.

Business Services

A business service is used to expose a back-end service so that it may be invoked by a script or a zone or a map to retrieve information or perform functions, depending on the related service.

As with the business object, the business service's interface to the internal service is defined using its schema. The schema maps the business service's elements to the corresponding elements in the internal service program's XML document. Just as a business object can simplify the schema of its maintenance object by only defining elements that it needs and “flattening” entries in a child collection to be defined as a singular element, a business service schema may simplify its service XML in a similar way.

Fastpath: Refer to Schema Syntax for a complete list of the XML nodes and attributes available to you when you construct a schema.

Inbound web services and scripts support interaction with business services. You can also invoke a business service from a Java class.

Service Program

This transaction defines services available in the system. These include user interface services as well as stand-alone services that perform a specific function. A service may be referenced by a business service. Use this transaction to view existing service and introduce a new stand-alone service to be made available to a Business Service.

Select **Admin** > **System** > **Service Program** to maintain service programs.

Description of Page

Service Name is the unique identifier of the service.

CAUTION: Important! When adding new service programs, carefully consider its naming convention. Refer to System Data Naming Convention for more information.

Owner indicates if this service is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a service. This information is display-only.

Description describes the service.

Application Service is the application service that is used to provide security for the service. If the service is related to a maintenance object, the access modes for the application service should be the standard **Add**, **Change**, **Delete** and **Inquire**. For other services, the application service should have the **Execute** access mode.

Service Type indicates whether the service is a **Java Based Service** or a **Java (Converted) Service**. Note that services generated to support a portal in the system will not have a service type populated.

Copyright © 2024, Oracle and/or its affiliates.

This **Program Component** grid shows the list of program user interface components associated with the service. For a stand-alone service, this list is typically not applicable.

Defining Business Services

The topics in this section describe how to maintain business services.

Note that several context sensitive sidebar zones appear on this page and are visible on all tabs.

- **Schema Tips**. This zone provides several links to launch help topics related to valid schema syntax.
- **Generate Schema**. This zone includes a button that can be used to generate the schema based on the XML of its related Service. Once generated, adjust the schema as desired.

Business Service - Main

Use this page to define basic information about a Business Service. Open this page using **Admin** > **System** > **Business Service**

Use the **Business Service Query** portal to search for an existing business service. Once a business service is selected, you are brought to the maintenance portal to view and maintain the selected record.

Description of Page

Enter a unique **Business Service** name and **Description**. Use the **Detailed Description** to describe the purpose of this business service in detail. **Owner** indicates if the business service is owned by the base package or by your implementation (**Customer Modification**).

> **CAUTION:** Important! If you introduce a new business service, carefully consider its naming convention. Refer to System Data Naming Convention for more information.

Enter the internal **Service Name** being called when this business service is invoked.

Enter the **Application Service** that is used to provide security for the business service. The application service must have an Access Mode of Execute.

Click the **View Schema** to view the business service's expanded schema definition. Doing this opens the schema viewer window.

Click the **View XSD** hyperlink to view the business object's expanded schema definition in XSD format.

Business Service - Schema

Use this page to maintain a Business Service's schema and to see where the Business Service is used in the system. Open this page using **Admin** > **System** > **Business Service** and then navigate to the **Schema** tab.

Description of Page

The contents of this section describe the zones that are available on this portal.

The **General Information** zone displays main attributes of the business service.

The Schema Designer zone allows you to edit the business service's schema. The purpose of the schema is to map the business service's elements to the corresponding fields of the back-end service program it rides on.

> **Note: Generating a Schema** A context sensitive "Generate Schema" zone is associated with this page. The zone provides a button that allows the user to generate a basic schema that includes all the elements that are found in the XML of the BS's service.

> **Fastpath:** Refer to Schema Syntax for a complete list of the XML nodes and attributes available to you when you construct a schema. Also note that the **Schema Tips** zone in the sidebar provides a link to launch this help topic directly.

Copyright © 2024, Oracle and/or its affiliates.

The **Schema Usage Tree** zone summarizes all cross-references to this schema. These may be other schemas, scripts, and web services. For each type of referencing entity, the tree displays a summary node showing a total count of referencing items. The summary node appears if at least one referencing item exists. Expand the node to list the referencing items and use their description to navigate to their corresponding pages.

Useful Services and Business Services

The following section highlights some business services and services provided by the product that may be useful for implementations to use.

Data Explorer Service

The system provides a mechanism for performing an SQL select statement for use in scripting, Java plug-ins, or via a web service call. This is done by creating a zone using one of the data explorer zone types where the SQL is defined. Then, create a business service using the Data Explorer service (**FWLZDEXP**).

Note: There are numerous business services delivered with the base product that reference this service that may be used as a template.

The following points highlight how to create your own business service for this service. Note that typically a separate business service exists for each zone.

- Enter a **Business Service** code and a **Description**. It is recommended to define the business service code to match the zone code so that it's easier to manage which business service invokes which zone.
- Select the **Service Name FWLZDEXP**.
- On the **Schema** tab, under the ***<schema>*** node, enter mapping for the fields that are required for the Data Explorer service:
  - The **Zone** should be mapped into service field ***ZONE_CD*** . Define the zone code as the default value.
  - For every **user filter** defined on the zone, create a schema mapping into the service field ***Fx_VALUE***, where "x" is the filter number (from the zone parameters).
  - For every **hidden filter** defined on the zone, create a mapping into the service field ***Hx_VALUE***, where "x" is the filter number (from the zone parameters).
  - The search results are returned as a list by the data explorer service. Each column value is in the service field ***COL_VALUE*** with an appropriate sequence number (***SEQNO***). The results can be flattened based on sequence number allowing for a logical element name to be defined.
  - Another useful field is ***ROW_CNT***, which provides the number of rows retrieved by your search.

The following is an example of the schema for a BS that receives a business object code and returns a list of status values and their descriptions that allow status reasons to be defined.

```
<schema>
     <zone mapField="ZONE_CD" default="F1-BOSTSLST"/>
     <bo mapField="H1_VALUE" -/>>
     <rowCount mapField="ROW_CNT"/>>
     <results type="list" mapList="DE">>
          <status dataType="string" mapField="COL_VALUE">
               <row mapList="DE_VAL">>
                    <SEQNO is="1" -/>>
               </row>>
          </status>>
          <description dataType="string" mapField="COL_VALUE">
               <row mapList="DE_VAL">>
                    <SEQNO is="2" -/>>
               </row>>
          </description>>
      </results>>
</schema>
```

Copyright © 2024, Oracle and/or its affiliates.

Maintenance Object Log Service

Many maintenance objects support a log table that follows a pattern of column names and behavior. The system provides a service called Generic MO Log Service (**F1MOLOGP**) that may be used to perform common functions related to log entries:

- Read log entries. If you pass a certain MO, primary key and log sequence number, the service will return the details of that log entry. The product provides a generic business service that may be used for this purpose - Generic MO - Retrieve Log Details (**F1–ReadMOLog**). Alternatively, it is possible to create a business service for a given MO where the MO code is assigned to the MO element using the default syntax. This allows business functionality specific to that maintenance object to use the specific BS.
- Add log. The service may be used to add a log entry. If a user log is added, then the comments from the user are populated in the detailed description. System generated log entries typically supply a message category / message number along with other information such as the status, a specific log type and optionally a related object reference (via a characteristic). The product provides a generic business service that may be used for this purpose - Add Generic MO Log (**F1–AddMOLog**). Alternatively, it is possible to create a business service for a given MO where the MO code is assigned to the MO element using the default syntax. This allows business functionality specific to that maintenance object to use the specific BS.

Base Business Services

The following table highlights some business services provided by the product that may be useful for custom logic for an implementation.

CAUTION: This is not intended to be a complete reference of Business Services. Refer to the business service page to find all the supported business services.

Business Object Related Services

Business Service Name	Description
F1-AutoTransitionBO	Performs monitoring algorithms associated with the current state of a given business object instance (which may result in subsequent state transitioning).
F1-CompareBusinessObjectData	Compares two versions of a given business object instance.
F1-DetermineBo	Determines the business object of a given instance of a maintenance object by executing the MO's Determine BO logic.
F1-GetRequiredFieldsForBOState	Returns the required fields for a given business object status.
F1-RetrieveBOOption	Returns BO option values for a given BO and option type.
F1-RetrieveBOStatusOption	Returns BO option values for a given BO, status and option type.
F1-RetrieveBOStatusOption	Retrieves a list of BOs for a given MO that are accessible for the current user.
F1-RetrieveBoStatusDescription	Return the description of a given BO status.
F1-RetrieveBusinessObjectLabel	Return the label appropriate for a given path (e.g. element) within a BO schema.
F1-RetrieveNextStates	Return a list of next possible states based on the input of a MO and its prime key, or a BO and one of its statuses.

Email Related Services

Copyright © 2024, Oracle and/or its affiliates.

Business Service Name	Description
F1-EmailService	Sends an email message in real time.
F1-RetrieveEmailAddress	Retrieves the email addresses of users belonging to a To Do Role.
F1-RetrieveEnvironmentURL	Retrieves the current environment URL information for the installation.

Tools for Maps and Scripting

Business Service Name	Description
F1-AddWarning	This business service may be used to log a warning by providing the message details. The system gathers all the warning and displays them to the user all at once.
F1-CheckOnlineConnection	Check if Connection is Online or Batch.
F1-CheckWarning	Used to check if any warnings have been accumulated. If warnings have been accumulated, any logic that cannot be rolled back (when the warnings are shown) should not be performed.
F1-DateMath	Performs various date and time math calculations. Refer to the BS description for more details.
F1-DateTimeFormattingService	Formats a given date / time based on the user's display profile settings.
F1-ExecuteScriptInNewSession	Executes a Service Script in a new processing session/ transaction.
F1-GetFieldLabel	Retrieves the label for a given field.
F1-GetForeignKeyReference	Returns foreign key reference information for a given FK Reference and primary key, including info description, navigation option, and context menu.
F1-GetFKReferenceDetails	Returns foreign key reference information for a given MO and primary key, including FK reference code, info description, navigation option, search zone and context menu.
F1-GetLookupDescription	Returns lookup description for a lookup field value given the lookup field name.
F1-GetExtLookUpVal	Returns the list of values for a given extendable lookup BO.
F1-GetMonthInYearAbbreviation	Returns a 3-character month abbreviation for an input date in system format.
F1-NumberAmountFormatter	Formats a given amount or number based on the user's display profile settings. It also may receive input to adjust the scale and optionally apply currency settings.
F1-OutmsgDispatcher	Dispatches a real-time message giving the user the option of whether to persist the message on the database, and whether to trap errors that may take place during the call. Refer to Real Time Messages for more information.
F1-OutmsgMediator	Alternative to **F1-OutmsgDispatcher** and may be a better option if the sender does not require an outbound message

Copyright © 2024, Oracle and/or its affiliates.

Business Service Name	Description
	record to be instantiated. Refer to Real Time Messages for more information.
F1-RethrowError	Issues an application error using the input message category / number / parameters.
F1-RetrieveMODescription	Retrieves the description for a maintenance object.
F1-ReturnMessage	Returns the expanded message given a message category, number, parameters, and parameter types.
F1-SavePointDispatcher	Allows for a service script to be executed where exceptions are trapped and the transaction is rolled back to a save point set before the service script execution.

User Related Services

Business Service Name	Comments
F1-CheckApplicationSecurity	Checks a user's security for a given application service / access mode
F1-CheckUserAuthorization	Determine whether a given user is authorized for access based on the input application service, security code, and authorization level.
F1-DetermineIfUserCanApproveTD	Determine if the current user can approve a given To Do.

User Interface (UI) Maps

The User Interface (UI) map holds HTML to be rendered within portal zones and Business Process Assistant (BPA) scripts. UI maps allow your implementation to create input forms and output maps that closely match your customer's business practices. In other words, the UI Map is designed to facilitate the capture and display of your business objects and business services.

The UI map is a repository for a single HTML document paired with an XML schema where the schema defines the data that the HTML document displays and/or modifies. The UI Map HTML gives you the ability to craft the display by any method that an html document can support, including JavaScript and full CSS functionality.

Configuration tool support for UI Maps hinges around the ability to inject and extract an XML document from the HTML. For more information on the specialized support for HTML and JavaScript functionality refer to UI Map Attributes and Functions.

Copyright © 2024, Oracle and/or its affiliates.

```
<html>
<head>
<title>Output Personal Information</title>
<link rel="stylesheet" type="text/css" href="cm_templates/cmStyles.css"/>
</head>
<body>

<table width="100%" border="0" cellpadding="2" style="margin-top:15px;" >

    <tr>
        <td/>
         <td/> <!-- locate the edit button on the bottom of the third column -->
        <td rowspan="99" style="vertical-align:bottom; margin-left:5px">
            <input type="button" value="edit" onClick="oraRunScript('HumanInfoU','personId');"/>
        </td>
    </tr>

    <tr>
        <td class="outputLabel">Name:</td>
        <td><span oraField="name" class="outputData"></span></td>
    </tr>
    <tr>
        <td class="outputLabel">Home Phone:</td>
        <td><span oraField="homePhone" class="outputData"></span></td>
    </tr>
    <tr>
        <td class="outputLabel">Business Phone:</td>
        <td><span oraField="businessPhone" class="outputData"></span></td>
    </tr>
    <tr>
        <td class="outputLabel">Cell Phone:</td>
        <td><span oraField="cellPhone" class="outputData"></span></td>
    </tr>
    <tr>
        <td class="outputLabel">FAX:</td>
        <td><span oraField="fax" class="outputData"></span></td>
    </tr>
    <tr>
        <td class="outputLabel">Social Security:</td>
        <td><span oraField="socialSecurity" class="outputData"></span></td>
    </tr>
    <tr>
        <td class="outputLabel">Drivers License:</td>
        <td><span oraField="driversLicense" class="outputData"></span></td>
    </tr>
    <tr>
        <td class="outputLabel">Email:</td>
        <td><span oraField="email" class="outputData"></span></td>
    </tr>

</table>

</body>

<xml>
<root>
    <name>Greer,Johan</name>
    <email>jurgen.greer@media.com</email>
    <socialSecurity>939-30-3939</socialSecurity>
    <driversLicense>C8392020</driversLicense>
    <homePhone>(838) 030-0303</homePhone>
    <cellPhone>(444) 444-4040</cellPhone>
    <businessPhone>(737) 393-3838</businessPhone>
    <fax>(373) 939-3939</fax>
    <personId>1239997654</personId>
</root>
</xml>
</html>
```

Figure 1: HTML to Display Customer Business Object

Name:	**Greer,Johan**
Home Phone:	**(838) 030-0303**
Business Phone:	**(737) 393-3838**
Cell Phone:	**(444) 444-4040**
FAX:	**(373) 939-3939**
Social Security:	**939-30-3939**
Drivers License:	**C8392020**
Email:	**jurgen.greer@media.com** edit

Figure 2: Customer HTML Rendered (Output Data for Zone)

Copyright © 2024, Oracle and/or its affiliates.

UI maps are typically crafted as output tables when used in conjunction with portal zones - please refer to Map Zones for more information. When referenced within BPA scripts, UI maps are typically crafted as forms for the capture and update of data.

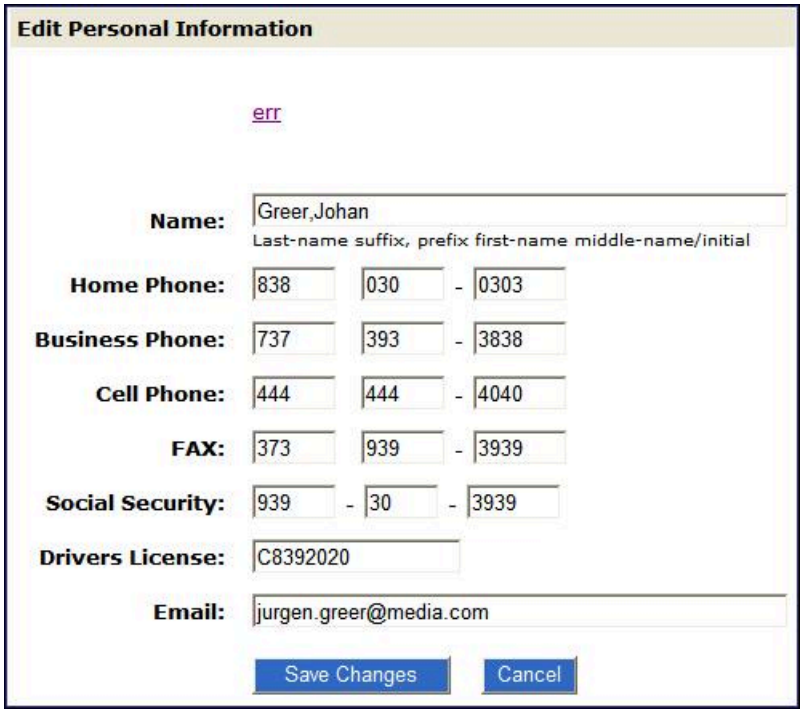


Figure 3: HTML Input Form Rendered (for BPA Script)

Portal zones can reference a UI map for the zone header. They may also utilize a UI map to define their filter area. This type of UI map is not a complete HTML document, but is instead configured as a UI Map "fragment".

Note: UI Map Tips. A context sensitive "UI Map Tips" zone is visible on the UI map maintenance page. This zone provides several links to launch help topics related to valid schema syntax and UI Hint syntax in one click.

Note: Editing HTML. You can use a variety of HTML editors to compose your HTML, which you can then cut, and paste into your UI map. In addition, the zone provides a complete list of the XML schema nodes and attributes available to you when you construct the map's data schema.

Defining UI Maps

The topics in this section describe how to maintain UI Maps.

UI Map - Main

Use this page to define basic information about a user interface (UI) Map. Open this page using **Admin** > **System** > **UI Map**

Use the **UI Map Query** portal to search for an existing UI Map. Once a map is selected, you are brought to the maintenance portal to view and maintain the selected record.

Description of Page

Enter a unique **Map** name. **Owner** indicates if the UI map is owned by the base package or by your implementation (**Customer Modification**).

Copyright © 2024, Oracle and/or its affiliates.

> **CAUTION:** Important! If you introduce a new UI map, carefully consider its naming convention. Refer to System Data Naming Convention for more information.

Use **UI Map Type** to indicate whether the map is a **Complete XHTML Document**, **Complete HTML Document**, **XHTML Fragment** or an **HTML Fragment**. The default value is **Complete XHTML Document**.

Enter a **Description**. Use the **Detailed Description** to describe how this map is used in detail.

Click on the **View Schema** to view the UI map's expanded schema definition. Doing this opens the schema viewer window.

Use the **Test UI Map** hyperlink to render your HTML in a test window.

> **Note:** The **Test UI Map** hyperlink also exercises the proprietary functionality that binds an XML element with an HTML element so you can get immediate feedback on your HTML syntax.

UI Map - Schema

Use this page to maintain a UI Map's HTML and schema and to see where the UI Map is used in the system. Open this page using **Admin** > **System** > **UI Map** and then navigate to the **Schema** tab.

Description of Page

The contents of this section describe the zones that are available on this portal.

The **General Information** zone displays main attributes of the UI Map.

The **HTML Editor** zone allows you to edit the HTML document of the map.

> **Note:** Refer to UI Map Attributes and Functions and UI Map Standards for more information about HTML definition syntax. These topics describe good ways to produce simple HTML, however, they are not an HTML reference. Note that you can use a variety of HTML editors to compose your HTML, which you can then cut and paste into your UI map.

> **Note: Providing Help.** A tool tip can be used to display additional help information to the user. This applies to section elements as well as individual elements on a map. Refer to UI Map Attributes and Functions for more information on how to enable and provide UI map help.

The **Schema Designer** zone allows you to edit the data schema part of the map. The purpose of the schema is to describe the data elements being displayed by the map.

> **Fastpath:** Refer to Schema Syntax for a complete list of the XML nodes and attributes available to you when you construct a schema. Also note that the **UI Map Tips** zone in the sidebar provides a link to launch this help topic directly.

The **Schema Usage Tree** zone summarizes all cross-references to this schema. These may be other schemas, scripts, and web services. For each type of referencing entity, the tree displays a summary node showing a total count of referencing items. The summary node appears if at least one referencing item exists. Expand the node to list the referencing items and use their description to navigate to their corresponding pages.

UI Map Attributes and Functions

> **Note:** This topic uses the term "field" to refer to both the generic concept of displaying and capturing data in a 'field' as well as referring to the meta-data object supplied in the product to define Fields. When referring to the latter, the term "MD Field" (meta-data Field) is used and a hyperlink to the Field documentation is provided.

Bind XML to HTML

Only two different attributes are required to bind a UI Map's XML to its HTML. Both of these attributes require an XML document embedded within the HTML, where the XML is bounded by <xml> nodes.

> **Warning:** You must embed a pair of <xml></xml> tags within your HTML document for binding to occur.

Copyright © 2024, Oracle and/or its affiliates.

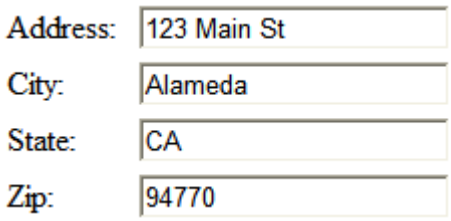
Linking a Field

Syntax	Values	Description
oraField=" "	Field element XPath	This attribute is used to link an HTML element directly with an XML element, where the XML element is defined within the UI Map's XML schema. The attribute can be used with any *rendering* HTML element, such as: , , and <input>.

- HTML for input element:

```
<html>
<body>
<table>
   <tr>
      <td>Address:</td>
      <td><input type="text" oraField="address"/></td>
   </tr>
   <tr>
      <td>City:</td>
      <td><input type="text" oraField="city"/></td>
   </tr>
   <tr>
      <td>State:</td>
      <td><input type="text" oraField="state"/></td>
   </tr>
   <tr>
      <td>Zip:</td>
      <td><input type="text" oraField="zip"/></td>
   </tr>
</table>
</body>
<xml>
   <root>
      <address>123 Main St</address>
      <city>Alameda</city>
      <state>CA</state>
      <zip>94770</zip>
   </root>
</xml>
</html>
```

Rendered HTML



- HTML for span and div elements:

```
<html>
<body>
 -
<div oraField="address"></div>
<span oraField="city"></span>
<span>,</span>
<span oraField="state"></span>
<span oraField="zip"></span>
<span oraField="country"></span>
 -
</body>
```

Copyright © 2024, Oracle and/or its affiliates.

```
<xml>
   <root>
      <address>123 Main St</address>
      <city>Alameda</city>
      <state>CA</state>
      <zip>94770</zip>
   </root>
</xml>
</html>
```

HTML rendered:

123 Main St
Alameda, CA 94770

Linking a List

This attribute is used to link an HTML table with an XML list, where the XML list is defined within the UI Map's XML schema. The purpose of the element is to trigger the framework to replicate the table's HTML for each occurrence of the list.

Syntax	**Values**	**Description**
oraList=" "	List element XPath	This attribute is used to link an HTML table with an XML list, where the XML list is defined within the UI Map's XML schema. The purpose of the element is to trigger the framework to replicate the table's HTML for each occurrence of the list.

Note: The oraField attributes embedded within the list must contain XPath navigation relative to the list. See below for an example.

```
<html>
<head><title>Bind xml list element</title></head>
<body>
<table oraList="payment">
   <thead>
      <tr>
         <th><span>Pay Date</span></th>
         <th><span>Amount</span></th>
        </tr>
        <tr/>
    </thead>
    <tr>
      <td>
         <span oraField="date" oraType="date"></span>
      </td>
      <td align="right">
         <span oraField="amount" oraType="money"></span>
      </td>
   </tr>
</table>
</body>
<xml>
<root>
    <payment>
        <date>2008-01-01</date>
        <amount>44.28</amount>
    </payment>
    <payment>
        <date>2008-02-01</date>
        <amount>32.87</amount>
    </payment>
    <payment>
        <date>2008-03-01</date>
```

Copyright © 2024, Oracle and/or its affiliates.

```
        <amount>21.76</amount>
    </payment>
</root>
</xml>
</html>
```

HTML rendered:

Pay Date	Amount
01-01-2008	$44.28
02-01-2008	$32.87
03-01-2008	$21.76

Build a Dropdown List

The following attributes are provided to build an HTML ‘select’ element, also called a dropdown, based on various sources.

Source	Syntax	Values	Examples

Lookup	oraSelect="lookup: ;"	Lookup field	`...`
			`<td>House Type:</td>`
			`<td>`
			`  <select - oraField="houseType" - oraSelect="lookup:HOUSE_TYPE;">`
			`</select>`
			`</td>`

Extendable Lookup	oraSelect="lookupBO: ;"	BO code	`...`
			`<td>UI Device Display Type:</td>`
			`<td>`
			`  <select - oraField="uiDeviceType" - oraSelect="lookupBO: F1-DeviceDisplayTypes;">`
			`</select>`
			`</td>`

Characteristic Type (pre-defined)	oraSelect="charType: ;"	Characteristic Type code	`...`
			`<td>Usage:</td>`
			`<td>`
			`  <select - oraField="statusReasonUsage" - oraSelect="charType:F1-SRUSG;">`
			`</select>`
			`</td>`
			`...`

Control Table	oraSelect="table: ;"	Table name	`...`
			`<td>Currency: </td>`
			`<td>`

Copyright © 2024, Oracle and/or its affiliates.

Source	Syntax	Values	Examples

	Note: This attribute only works with tables that follow the standard control table structure where there is a related language table that includes that includes the column **DESCR** as its description column. Use the Application Viewer data dictionary to identify tables that qualify for this functionality.
	
	Warning: The oraSelect function will only work if less than 500 values are displayed.
			`<select - oraField="currency" - oraSelect="table:CI_CURRENCY_CD;">`
			`</select>`
			`</td>`
			`...`

Page Service	oraSelect="service: ;"	Page Service name	`...`
			`<td>Country:</td>`
			`<td>`
			`<select - oraField="country" - oraSelect="service:CIPTCNTW;">`
			`</select>`
			`</td>`
			`...`

Embedded List	Used to build a select dropdown based on a list within the map's XML.
	
	oraSelect="valuePath: ;descPath: "
		After the valuePath, indicate the XPath of the element that holds the values. After the descPath, indicate the XPath of the element that holds the descriptions.	`<html>`
			`<body>`
			`<table summary="" border="0" - cellpadding="1" - cellspacing="1">`
			`<tr>`
			`<td>Select: </td>`
			`<td><select oraSelect= "valuePath:list/value; - descPath:list/desc" - oraField="target"></select></td>`
			`</tr>`
			`</table>`
			`</body>`
			`<xml>`
			`<root>`
			`<target>10</target>`
			`<list>`
			`<value>10</value>`
			`<desc>Ten</desc>`
			`</list>`
			`<list>`
			`<value>20</value>`
			`<desc>Twenty</desc>`
			`</list>`
			`<list>`
			`<value>40</value>`
			`<desc>Forty</desc>`
			`</list>`
			`</root>`
			`</xml>`

Copyright © 2024, Oracle and/or its affiliates.

Source	Syntax	Values	Examples
			`</html>`
Service Script	oraSelect="ss: ;"	Service Script code	See below for additional syntax needed when using this function.
Business Service	oraSelect="bs: ;"	Business Service code	See below for additional syntax needed when using this function.

When specifying a service script or a business service, extra mapping information is needed to pass data to and from the service.

Syntax	Values	Description
oraSelectIn=" ;"	serviceXPath:element;	Used to pass the value of another element into the service (mapping to the service's XPath).
	serviceXPath:'Literal';	Used to pass a constant or literal to the service (mapping to the service's XPath).
oraSelectOut="valuePath: ; descPath: "	See examples below	Used to indicate which element in the service's output holds the values and which one holds the descriptions.

Example using a business service:

```
...
<td>External System: </td>
<td>
   <select oraField="externalSystem" -
    oraSelect="bs:F1-RetrieveExternalSystems" -
    oraSelectIn="outboundMsgType:boGroup/parameters/outboundMsgType;" -
    oraSelectOut="valuePath:results/externalSystem; -
    descPath:results/description">
   </select>
</td>
...
```

This method for building dropdowns is often used when there is a dependency between elements and the list of valid values in a dropdown (for the child element) is based on another element in the map (the parent element). When the parent element is changed, it may be required to refresh the child element. This behavior can be implemented using the function called within an **onChange** event in the map. The syntax is **oraHandleDependentElements('dependent element');**. Multiple target elements (dependents) can be named.

The following example is related to the above business service example where the list of external systems is specific for a given outbound message type, which is passed in as input. The snippet below shows the configuration for the outbound message type element to trigger a refresh of the external system’s dropdown list.

```
...
<div>
   <label oraLabel="boGroup/parameters/outboundMsgType"></label>
      <span>
         <select oraSelect="table:F1_OUTMSG_TYPE"  -
          oraField="boGroup/parameters/outboundMsgType"
          onChange="oraHandleDependentElements('boGroup/parameters/
externalSystem');">
         </select>
      </span>
</div>
...
```

Copyright © 2024, Oracle and/or its affiliates.

Format Input and Output Fields

The following attributes are designed to apply data type formatting for input and output fields.

Automatic Formatting

Syntax
oraSchemaDataTypes="false"

This attribute is used to trigger automatic formatting in the rendered HTML document. The automated formatting will occur according to the data type attributes defined in the UI map's schema. For details on specific data type formatting, please refer to the oraType attribute descriptions below.

Warning: The attribute **oraSchemaDataTypes="true"** will be automatically injected into the UI map's HTML! If you do not wish to apply the schema's data types to the rendered HTML then you must specify this attribute in the body node with a value of false. The attribute **<body oraSchemaDataTypes="false">** is required to avoid automatic formatting!

- UI Map schema:

```
<schema>
    <schemaDate dataType="date"/> -
    <schemaDateTime dataType="dateTime"/> -
    <schemaFKRef fkRef="CI_USER"/> -
    <schemaLookup dataType="lookup" lookup="ACCESS_MODE"/> -
    <schemaMoney dataType="money"/> -
    <schemaNumber dataType="number"/> -
    <schemaTime dataType="time"/> -
</schema>
```

- UI Map HTML:

```
<html>
<body oraSchemaDataTypes="true">
<table border="1" cellpadding="1" cellspacing="1">
   <tr><th>dataType</th><th>result type</th><th>input result</th><th> display-
only result</th></tr>
   <tr>
      <td rowspan="2">date (from schema)</td>
      <td>raw</td>
      <td><input oraField="schemaDate" oraType="string" -/></td>
      <td><span oraField="schemaDate" oraType="string"></span></td>
   </tr>
   <tr>
      <td>rendered</td>
      <td><input oraField="schemaDate"></td>
      <td><span oraField="schemaDate"></span></td>
   </tr>
 -
   <tr>
      <td rowspan="2">dateTime (from schema)</td>
      <td>raw</td>
      <td><input oraField="schemaDateTime" oraType="string"></td>
      <td><span oraField="schemaDateTime" oraType="string"></span></td>
   </tr>
   <tr>
      <td>rendered</td>
      <td><input oraField="schemaDateTime"></td>
      <td><span oraField="schemaDateTime"></span></td>
   </tr>

   <tr>
      <td rowspan="2">fkRef (from schema)**</td>
      <td>raw</td>
      <td><input oraField="schemaFkRef" oraType="string"></td>
      <td><span oraField="schemaFkRef" oraType="string"></span></td>
   </tr>
```

Copyright © 2024, Oracle and/or its affiliates.

```
   <tr>
      <td>rendered</td>
      <td><input oraField="schemaFkRef"></td>
      <td><span oraField="schemaFkRef"></span></td>
   </tr>
 -
   <tr>
      <td rowspan="2">lookup (from schema)</td>
      <td>raw</td>
      <td><input oraField="schemaLookup" oraType="string"></td>
      <td><span oraField="schemaLookup" oraType="string"></span></td>
   </tr>
   <tr>
      <td>rendered</td>
      <td><input oraField="schemaLookup"></td>
      <td><span oraField="schemaLookup"></span></td>
   </tr>
   -
   <tr>
      <td rowspan="2">money (from schema)</td>
      <td>raw</td>
      <td><input oraField="schemaMoney" oraType="string"></td>
      <td><span oraField="schemaMoney" oraType="string"></span></td>
   </tr>
   <tr>
      <td>rendered</td>
      <td><input oraField="schemaMoney"></td>
      <td><span oraField="schemaMoney"></span></td>
   </tr>
 -
   <tr>
      <td rowspan="2">number (from schema)</td>
      <td>raw</td>
      <td><input oraField="schemaNumber" oraType="string"/></td>
      <td><span oraField="schemaNumber" oraType="string"></span></td>
   </tr>
   <tr>
      <td>rendered</td>
      <td><input oraField="schemaNumber"></td>
      <td><span oraField="schemaNumber"></span></td>
   </tr>
 -
   <tr>
      <td rowspan="2">time (from schema)</td>
      <td>raw</td>
      <td><input oraField="schemaTime" oraType="string"></span></td>
      <td><span oraField="schemaTime" oraType="string"></span></td>
   </tr>
   <tr>
      <td>rendered</td>
      <td><input oraField="schemaTime"></td>
      <td><span oraField="schemaTime"></span></td>
   </tr>
 -
</table>
 -
</body>
<xml>
<root>
   <schemaDate>2007-11-02</schemaDate>
   <schemaDateTime>2007-11-02-23.45.00</schemaDateTime>
   <schemaFkRef>USD</schemaFkRef>
   <schemaLookup>A</schemaLookup>
   <schemaMoney>1000000</schemaMoney>
   <schemaNumber>5661976.11548</schemaNumber>
   <schemaTime>23.45.00</schemaTime>
</root>
</xml>
</html>
```

Copyright © 2024, Oracle and/or its affiliates.

HTML rendered.

dataType	result type	input result		display-only result
date	raw	2007-11-02		2007-11-02
	rendered	11-02-2007		11-02-2007
dateTime	raw	2007-11-02-23.45.00		2007-11-02-23.45.00
	rendered	11-02-2007	11:45PM	11-02-2007 11:45PM
fkRef	raw	SYSUSER		SYSUSER
	rendered	SYSUSER		SYSUSER
lookup	raw	A		A
	rendered	Add		Add
money	raw	1000000		1000000
	rendered	$1,000,000.00		$1,000,000.00
number	raw	5661976.11548		5661976.11548
	rendered	5,661,976.11548		5,661,976.11548
time	raw	23.45.00		23.45.00
	rendered	11:45PM		11:45PM

Date Formatting

This function is used to display a date according to the user's display profile. For input fields, the setting formats the data when the user tabs out of the field.

Syntax
oraType="date"

```
<html>
<body>
<table summary="" border="0" cellpadding="1" cellspacing="1">
   <tr>
      <td>Date: </td>
      <td><span oraField="date" oraType="date"></span></td>
   </tr>
   <tr>
      <td>Date: </td>
      <td><input oraField="date" oraType="date"/></td>
   </tr>
</table>
</body>
<xml>
<root>
   <date>2008-12-28</date>
</root>
</xml>
</html>
```

HTML rendered.

Copyright © 2024, Oracle and/or its affiliates.

Date: 12-28-2008

Date: 12-28-2008

Time Formatting

This function is used to display a time according to the user's display profile. For input fields, the setting formats the data when the user tabs out of the field.

Syntax
oraType="time"

```
<html>
<body>
<table summary="" border="0" cellpadding="1" cellspacing="1">
   <tr>
      <td>Time: </td>
      <td><span oraField="time" oraType="time"></span></td>
   </tr>
   <tr>
      <td>Time: </td>
      <td><input oraField="time" oraType="time"/></td>
   </tr>
</table>
</body>
<xml>
<root>
   <time>00.28.54.389</time>
</root>
</xml>
</html>
```

HTML rendered.

Time: 12:28AM

Time: 12:28AM

Date and Time Formatting

This function is used to display a timestamp according to the user's display profile.If this function is used for an input element, it is broken into two pieces, one for date and one for time. Optionally, the time portion of the date time element can be suppressed using the attribute value 'time:suppress'.

Syntax
oraType="dateTime; time:suppress"

```
<html>
<body>
<table summary="" border="0" cellpadding="1" cellspacing="1">
   <tr>
      <td>Date time: </td>
      <td><span oraField="dateTime" oraType="dateTime"></span></td>
   </tr>
   <tr>
      <td>Date only: </td>
      <td><span oraField="dateTime" oraType="dateTime; time:suppress"></span></td>
   </tr>
   <tr>
      <td>Date time: </td>
      <td><input oraField="dateTime" oraType="dateTime"/></td>
   </tr>
   <tr>
```

Copyright © 2024, Oracle and/or its affiliates.

```
      <td>Date only: </td>
      <td><input oraField="dateTime" oraType="dateTime; time:suppress"/></td>
   </tr>
</table>
</body>
<xml>
<root>
   <dateTime>2009-11-01-00.28.54</dateTime>
</root>
</xml>
</html>
```

HTML rendered.

Date time: 11-01-2009 12:28AM

Date only: 11-01-2009

Date time: 11-01-2009 | 12:28AM

Date only: 11-01-2009

Date / Time Formatting with Standard Time

This true function is used to render a date / time element according to the daylight savings time schedule of the 'base' time zone. The 'base' time zone is specified on the Installation table and represents the database time zone. For input elements with this setting, all time entered is assumed to correspond with the daylight savings time schedule of the base time zone. If a time is entered that cannot be unambiguously translated to standard time, then the user will be required to provide a time zone label to indicate whether daylight savings time, or standard time, has been entered.

Syntax
oraType="dateTime; stdTime:true;"

```
<html>
<body>
<table summary="" border="0" cellpadding="1" cellspacing="1">
   <tr>
      <td>Date time: </td>
      <td><span oraField="dateTime" oraType="dateTime; stdTime:true;"></span></td>
   </tr>
   <tr>
      <td>Date time: </td>
      <td><input oraField="dateTime" oraType="dateTime; stdTime:true;"/></td>
   </tr>
</table>
</body> -
<xml>
<root>
   <dateTime>2009-11-01-00.28.54</dateTime>
</root>
</xml>
</html>
```

HTML rendered.

Note: The time zone label is displayed because 1:28 is ambiguous otherwise. Legally, November 1, 2009 1:28 AM occurs twice because daylight savings time (DST) is removed at 2:00 AM. With the stdTime function time zone labels are only displayed when required to clarify time overlaps.

Date time: 11-01-2009 01:28AM PDT

Date time: 11-01-2009 | 01:28AM PDT

Copyright © 2024, Oracle and/or its affiliates.

HTML rendered for the following day.

Date time: 11-02-2009 12:28AM

Date time:	11-02-2009	12:28AM

Date and Time Formatting with Time Zone Reference

Syntax	Valid Values	Description
oraType="dateTime; stdTimeRef: ;"	Reference an XPath after the colon.	This function is used to render a date / time element according to the daylight savings time schedule of a time zone whose XPath is referenced. Note that the time processed is assumed to have been stored in the standard time of the referenced time zone - so only daylight savings time shifting will execute - not time zone shifting.
oraType="dateTime; displayRef: ;"	Reference an XPath after the colon.	This function is similar to the stdTimeRef function, except that this function will execute time zone shifting in addition to daylight savings time shifting. To use displayRef correctly, only associate it with time zone elements that have been stored in the base time zone.

For input elements, all time entered is assumed to correspond with the daylight savings time schedule of the referenced time zone. If a time is entered that cannot be unambiguously translated to standard time, then the user will be required to provide a time zone label to indicate whether daylight savings time, or standard time, has been entered.

```
<html>
<body>
<table summary="" border="0" cellpadding="1" cellspacing="1">
   <tr>
      <td>Date time: </td>
      <td><span oraField="dateTime" oraType="dateTime; stdTimeRef:timeZone;"></
span></td>
   </tr>
   <tr>
      <td>Date time: </td>
      <td><input oraField="dateTime" oraType="dateTime; stdTimeRef:timeZone;"/></td>
   </tr>
</table>
</body>
<xml>
<root>
   <timeZone>US-EAST</timeZone>
   <dateTime>2009-11-01-00.28.54</dateTime>
</root>
</xml>
</html>
```

HTML rendered.

Note: The time zone label is always displayed for a referenced time zone.

Date time: 11-01-2009 01:28AM EDT

Date time:	11-01-2009	01:28AM EDT

HTML rendered for the following day.

Copyright © 2024, Oracle and/or its affiliates.

Date time: 11-02-2009 12:28AM EST

Date time: 11-02-2009 | 12:28AM EST

Duration Formatting

Syntax
oraType="duration"

This function is used to display time duration. For an input element, the value entered by the user is translated from minutes to hour and minutes as appropriate. For example, an entered value of '90', is converted to '00:01:30' when tabbing out of the input field.

```
<html>
<body>
<table summary="" border="0" cellpadding="1" cellspacing="1">
   <tr>
      <td>Duration: </td>
      <td><span oraField="duration" oraType="duration"></span></td>
   </tr>
   <tr>
      <td>Duration: </td>
      <td><input oraField="duration" oraType="duration"/></td>
   </tr>
</table>
</body>
<xml>
<root>
   <duration>90</duration>
</root>
</xml>
</html>
```

HTML rendered.

Duration: 00:01:30

Duration: 00:01:30

Day in Month Formatting

Syntax
oraType="dayInMonth"

This function is used to display a concatenated day and month.

```
<html>
<body>
<table summary="" border="0" cellpadding="1" cellspacing="1">
   <tr>
      <td>Day In Month: </td>
      <td><span oraField="dayMonth" oraType="dayInMonth"></span></td>
   </tr>
   <tr>
      <td>Day In Month: </td>
      <td><input oraField="dayMonth" oraType="dayInMonth"/></td>
   </tr>
</table>
</body>
<xml>
<root>
   <dayMonth>0228</dayMonth>
</root>
```

Copyright © 2024, Oracle and/or its affiliates.

```
</xml>
</html>
```

HTML rendered.

Day In Month: 02-28

Day In Month: 02-28

Month In Year Formatting

Syntax
oraType="monthInYear"

This function is used to display a concatenated month and year.

```
<html>
<body>
<table summary="" border="0" cellpadding="1" cellspacing="1">
   <tr>
      <td>Month In Year: </td>
      <td><span oraField="month" oraType="monthInYear"></span></td>
   </tr>
   <tr>
      <td>Month In Year: </td>
      <td><input oraField="month" oraType="monthInYear"/></td>
   </tr>
</table>
</body>
<xml>
<root>
   <month>200811</month>
</root>
</xml>
</html>
```

HTML rendered.

Month In Year: 11-2008

Month In Year: 11-2008

Monetary Formatting

This function is used to display a number as a monetary amount. See the table for configuration options with respect to the currency. For input elements, an error is issued if a non-numeric value is entered.

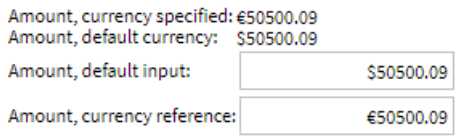
Syntax	**Description**
oraType="money: "	Directly specify a currency code after the colon.
oraType="money;currencyRef: "	Reference an XPath (after the colon) for an element that references a currency code.
oraType="money"	If no currency or currency reference is specified, the installation currency is used.

Copyright © 2024, Oracle and/or its affiliates.

Note: You must specify a pair of stylesheet references, cisEnabled and cisDisabled, in the map's header for right alignment. The stylesheet controls how the field will be rendered. If you want to alter the rendering you must override the oraMoney style.

```
<html>
<head>
    <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
    <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
   <tr>
      <td>Amount, currency specified:</td>
      <td><span oraType="money:EUR" oraField="totalAmt"></span></td>
   </tr>
   <tr>
      <td>Amount, default currency:</td>
      <td><span oraType="money" oraField="totalAmt"></span></td>
   </tr>
   <tr>
      <td>Amount, default input:</td>
      <td><input oraType="money" oraField="totalAmt"/></td>
   </tr>
   <tr>
      <td>Amount, currency reference:</td>
      <td><input oraType="money;currencyRef:cur" oraField="totalAmt"/></td>
   </tr>
</table>
</body>
<xml>
<root>
   <totalAmt>50500.09</totalAmt>
   <cur>EUR</cur>
</root>
</xml>
</html>
```

HTML rendered



Number Formatting

This function is used to display a number or validate an input value. For input elements, the system will return an error if a non-numeric value is entered.

Syntax
oraType="number"

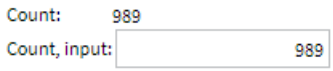
Note: You must specify a pair of stylesheet references, cisEnabled and cisDisabled, in the map's header for right alignment. The stylesheet controls how the field will be rendered. If you want to alter the rendering you must override the oraNumber style.

```
<html>
<head>
    <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
    <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
```

Copyright © 2024, Oracle and/or its affiliates.

```
   <tr>
      <td>Count:</td>
      <td><span oraType="number" oraField="count"></span></td>
   </tr>
   <tr>
      <td>Count, input:</td>
      <td><input oraType="number" oraField="count"/></td>
   </tr>
</table>
</body>
<xml>
<root>
   <count>989</count>
</root>
</xml>
</html>
```

HTML rendered.



FK Reference Formatting

By default, when an element with an **fkRef** oraType is displayed, an info string, context menu, navigation, and search are enabled (if the FK reference has been configured accordingly). Syntax is provided to allow you to selectively turn off any of these features.

Note that you can enable the foreign key hyperlink separately as well, refer to Embed Framework Navigation for more information. The various attributes used to control foreign key reference functionality are as follows. Note that in every case, the default value is **true**. A value of **false** should be used to disable the feature.

Syntax
oraType="fkRef:true|false; info:true|false; context:true|false; navigation:true|false; search:true|false"

- **fkRef**. A value of 'true' enables all of the following foreign key reference processing. Use a value of 'false' to disable automatic FK Reference processing.
- **info**. A value of 'true' renders an information string on the UI map, if applicable.
- **context**. A value of 'true' renders a context menu to appear before the foreign key reference element, if applicable.
- **navigation**. A value of 'true' causes the information string to be rendered as a hyperlink, if applicable. Clicking the hyperlink navigates to the appropriate page.
- **search**. A value of 'true' displays a search icon that launches the search zone if applicable.

Note: Foreign key navigation and context menu functionality is only available for UI maps presented in a portal zone. UI Maps presented during BPA script processing cannot support navigation options. Search functionality is only available for input HTML elements.

- UI Map schema:

```
<schema>
  <bo fkRef="F1-BOMO"/>
</schema>
```

- UI Map HTML:

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
   <tr>
      <td>Business Object</td>
```

Copyright © 2024, Oracle and/or its affiliates.

```
      <td><span oraField="bo" oraType="fkRef:true; info:true; context:true; -
navigation:true;"></span></td>
   </tr>
</table>
</body>
<xml>
<root>
   <bo>F1-COUNTRY</bo>
</root>
</xml>
</html>
```

- HTML rendered.

BUSINESS OBJECT Country

Lookup Formatting

This function is used to display the description of a lookup value.

Syntax	Valid Values
oraType="lookup: "	Lookup field name after the colon.

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
   <tr>
      <td>Status:</td>
      <td><span oraField="status" oraType="lookup:BATCH_JOB_STAT_FLG"></span></td>
   </tr>
</table>
</body>
<xml>
<root>
   <status>PD</status>
</root>
</xml>
</html>
```

HTML rendered.

Status:	Pending

Extendable Lookup Formatting

This function is used to display the description of an extendable lookup value.

Syntax	Valid Values
oraType="lookupBO: "	Business Object code name after the colon.

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
   <tr>
      <td>Value:</td>
      <td><span oraField="status" oraType="lookupBO:F1-DeviceDisplayTypes"></span></
td>
   </tr>
</table>
</body>
<xml>
<root>
    <status>oraTablet</status>
</root>
```

Copyright © 2024, Oracle and/or its affiliates.

```
</xml>
</html>
```

HTML rendered.

Value:	Tablet size

Characteristic Type Formatting

This function is used to display the description of a predefined characteristic value.

Syntax	Valid Values
oraType="charType: "	Characteristic Type code after the colon.

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
    <tr>
        <td>Skill:</td>
        <td><span oraType="charType:CM-SKILL" oraField="skill"></span></td>
    </tr>
</table>
</body>
<xml>
<root>
    <skill>10</skill>
</root>
</xml>
</html>
```

HTML rendered.

Skill:	Quality assurance

Control Table Formatting

This function is used to display the description of a control table that has an associated language table.

Syntax	Valid Values
oraType="table: "	Table code after the colon.

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
   <tr>
      <td>Currency:</td>
      <td><span oraType="table:CI_CURRENCY_CD" oraField="curr"></span></td>
   </tr>
</table>
</body>
<xml>
<root>
   <curr>USD</curr>
</root>
</xml>
</html>
```

HTML rendered.

Currency:	United States Dollars

Copyright © 2024, Oracle and/or its affiliates.

Add / Remove Grid Formatting

Syntax	Description
oraType="addGridRow"	The addGridRow function is used to build "insert row" dialog into the UI map. • An 'add' image is displayed. • Clicking the image inserts a new row in the grid. • If the list is empty, by default, an empty grid row is automatically be added. This means that the user will always see at least one grid row when this attribute is used.
oraType="deleteGridRow"	The deleteGridRow function is used to build "delete row" dialog into the UI map. • A 'delete' image is displayed. • Clicking the image removes the adjacent row from the grid.

Note: Because add and delete dialogs are relevant only inside a table, these attributes must be specified within a <td> element.

Warning: These attributes are designed to work with the business object action of 'replace' rather than 'update'. Therefore, if the map contains a grid, the business object action of 'replace' must be used to update the business object. Refer to BO Replace Action for more information.

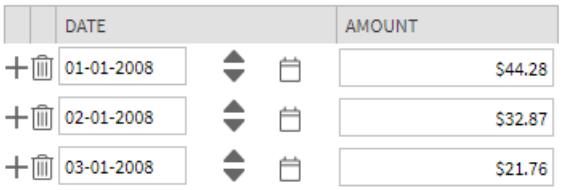
Example:

```
<html>
<head>
<title>Demonstrate Grid Add and Grid Delete OraTypes</title>
    <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
    <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body>
<table oraList="listEntry">
   <thead>
      <tr>
         <th/>
         <th/>
         <th><span>Date</span></th>
         <th><span>Amount</span></th>
      </tr>
      <tr/>
   </thead>
   <tr>
      <td oraType="addGridRow"></td>
      <td oraType="deleteGridRow"></td>
      <td>
         <input oraField="date" oraType="date"></input>
      </td>
      <td align="right">
         <input oraField="amount" oraType="money"></input>
      </td>
   </tr>
</table>
</body>
<xml>
<root>
   <listEntry>
      <date>2008-01-01</date>
      <amount>44.28</amount>
   </listEntry>
   <listEntry>
```

Copyright © 2024, Oracle and/or its affiliates.

```
      <date>2008-02-01</date>
      <amount>32.87</amount>
   </listEntry>
   <listEntry>
      <date>2008-03-01</date>
      <amount>21.76</amount>
   </listEntry>
</root>
</xml>
</html>
```

HTML rendered.



Unformatted Elements

This function is used to display the contents of an element that contains 'raw' data as defined for the schema element being rendered.

Syntax
oraType="raw"

- UI Map schema:

```
<schema>
    <rawStuff type="raw"/>
</schema>
```

- UI Map HTML:

```
<html>
<head>
   <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
   <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
   <tr>
      <td>Raw Stuff:</td>
      <td><span oraType="raw" oraField="rawStuff"></span></td>
   </tr>
</table>
</body>
<xml>
<root>
   <rawStuff>
      <ele1>text in element 1</ele1>
      <group1>
         <ele2>text inside element 2, group 1</ele2>
         <ele3>text inside element 3, group 1</ele3>
      </group1>
   </rawStuff>
</root>
</xml>
</html>
```

HTML rendered.

Copyright © 2024, Oracle and/or its affiliates.

Raw Stuff:	<ele1>text in element 1</ele1> ⊟ <group1> <ele2>text inside element 2, group 1</ele2> <ele3>text inside element 3, group 1</ele3> </group1>

String Formatting

This function is used to display the contents of an element, as XML pretty-print, when the element contains escaped XML.

Syntax
oraType="xmlString"

Note: It is not required, but the pretty print of the rendered XML is enhanced if you specify a pair of stylesheet references, cisEnabled and cisDisabled, in the map's header.

Example:

```
<html>
<head>
   <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
   <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
   <tr>
      <td>XML Stuff:</td>
      <td><span oraType="xmlString" oraField="xmlStuff"></span></td>
      </tr>
</table>
</body>
<xml>
<root>
   <xmlStuff>
      <ele1>text in element 1</ele1>
      <group1>
         <ele2>text inside element 2, group 1</ele2>
         <ele3>text inside element 3, group 1</ele3>
      </group1>
   </xmlStuff>
</root>
</xml>
</html>
```

HTML rendered.

XML Stuff:	<ele1>text in element 1</ele1> ⊟ <group1> <ele2>text inside element 2, group 1</ele2> <ele3>text inside element 3, group 1</ele3> </group1>

HTML rendered without oraType="xmlString"

XML Stuff:	<ele1>text in element 1</ele1><group1><ele2>text inside element 2, group 1</ele2><ele3>text inside element 3, group 1</ele3></group1>

Copyright © 2024, Oracle and/or its affiliates.

HTML Formatting

This function is used to display the contents of an element as HTML as opposed to plain text. An element defined as oraType="fkref" is automatically rendered as HTML.

Syntax
oraType="html"

Warning:

To avoid execution of malicious HTML not all HTML tags are supported. The list of supported tags is defined in the "F1-HTMLWhiteList" managed content definition.

If unsupported HTML is detected the entire element is escaped and rendered as plain text. It is therefore recommended to properly escape any source string that contributes to the final HTML element if it is not expected to contain valid HTML. This way only the offending string is escaped and not the entire element.

If the HTML element is composed in scripting refer to the 'escape' function described in the Edit Data Syntax for more information. Use the WebStringUtilities.asHTML java API for escaping text composed in Java.

```
<html>
<head>
   <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
   <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body>
<table summary="" border="0" cellpadding="1" cellspacing="1">
   <tr>
   <td>Info -:</td>
   <td><span oraType="html" oraField="info"></span></td>
   </tr>
</table>
</body>
<xml>
<root>
   <info><b>text in bold</b></info>
</root>
</xml>
</html>
```

HTML rendered.

Info : **text in bold**

HTML rendered without oraType="html"

Info : text in bold

Display Labels

Derive Label from an Element

This attribute is used to obtain a language sensitive label for a , <td>, or <input> HTML element.

Syntax	**Valid Values**
oraLabel=" "	XPath of an element in the UI map schema. The element must reference either the **mapField=**, **mdField=**, or **label=** attribute.

Copyright © 2024, Oracle and/or its affiliates.

Note: You can also define a field directly in your HTML for label definition, refer to Deriving Label from a Field for more information.

Note: If the schema contains multiple attributes, the oraLabel attribute will pick the label to render according to the following hierarchy: The label attribute overrides the mdField attribute, which in turn will override the mapField attribute.

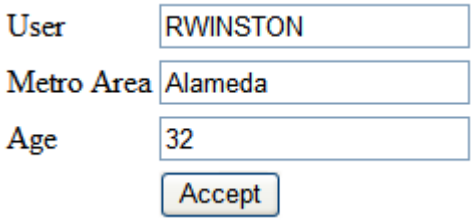
- UI Map schema:

```
<schema>
   <user mapField="USER_ID"/> -
   <info type="group" mapXML="CLOB">
      <city label="Metro Area"/> -
      <age mdField="AGE_LBL"/>
   </info> -
</schema>
```

- HTML:

```
<html>
<head><title oraMdLabel="BUS_LBL"></title></head>
<body>
<table>
   <tr>
      <td oraLabel="user"></td>
      <td><input oraField="user"/></td>
   </tr>
   <tr>
      <td oraLabel="info/city"></td>
      <td><input oraField="info/city"/></td>
   </tr>
   <tr>
      <td oraLabel="info/age"></td>
      <td><input oraField="info/age"/></td>
   </tr>
   <tr>
      <td/>
      <td><input type="button" oraMdLabel="ACCEPT_LBL"/></td>
   </tr>
</table>
</body>
<xml>
<root>
   <user>RWINSTON</user>
   <info>
      <city>Alameda</city>
      <age>32</age>
   </info>
</root>
</xml>
</html>
```

HTML rendered:



Deriving Label from a Field

Copyright © 2024, Oracle and/or its affiliates.

This attribute is used to obtain a language sensitive label for a , <td>, <input>, or <title> HTML element. The label text is derived from the field referenced.

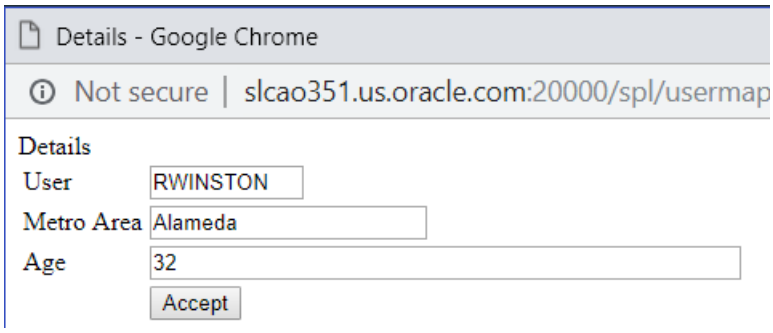
Syntax	Valid Values
oraMdLabel=" "	MD Field code.

Note: You can also define labels derived from the map's schema definition, refer to Derive Label from an Element for more information.

- HTML:

```
<html>
<head><title oraMdLabel="F1_DETAILS_LBL"></title></head>
<body>
<table>
   <tr>
      <td oraLabel="user"></td>
      <td><input oraField="user"/></td>
   </tr>
   <tr>
      <td oraLabel="info/city"></td>
      <td><input oraField="info/city"/></td>
   </tr>
   <tr>
      <td oraLabel="info/age"></td>
      <td><input oraField="info/age"/></td>
   </tr>
   <tr>
      <td/>
      <td><input type="button" oraMdLabel="ACCEPT_LBL"/></td>
   </tr>
</table>
</body>
<xml>
<root>
   <user>RWINSTON</user>
   <info>
      <city>Alameda</city>
      <age>32</age>
   </info>
</root>
</xml>
</html>
```

HTML rendered:



Copyright © 2024, Oracle and/or its affiliates.

Enable UI Map Help

The Display Labels section describes ways to derive the label for an element using an underlying MD Field. In addition, if the same MD field contains help text, the system will automatically generate a tool tip adjacent to the element label. Clicking the tool tip allows the user to view the help text.

It is possible to change the rendering of the tool tip. Refer to Custom Look And Feel Options for more information

Search Using a Pop-Up Explorer Zone

Search Option

This attribute is used to enable search zone functionality for input HTML elements.

Syntax	Valid Values
oraSearch=" "	Zone code.

Note: The oraSearch attribute is similar to the oraType attribute, because it will be 'automatically' included into HTML via the oraSchemaDataTypes attribute. This means that coding the oraSearch attribute into UI Map HTML is only required if a search zone has not been specified in the schema, or in the schema element's FK reference.

- Example of defining the search in the HTML. UI Map's Schema:

```
<schema>
   <uiMap/> -
</schema>
```

UI Map's HTML

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
   <tr>
      <td>UI Map with Search </td>
      <td><input oraField="uiMap" oraSearch="F1-UISRCH"></td>
   </tr>
</table>
</body>
<xml>
<root>
   <uiMap/>
</root>
</xml>
</html>
```

- Example of defining the search in the schema. UI Map's Schema:

```
<schema> -
   <uiMap search="F1-UISRCH"/>
</schema>
```

UI Map's HTML

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
   <tr>
      <td>UI Map with Search </td>
      <td><input oraField="uiMap"></td>
   </tr>
</table>
</body>
<xml>
<root>
```

Copyright © 2024, Oracle and/or its affiliates.

```
   <uiMap/>
</root>
</xml>
</html>
```

- Example where the FK reference defines the search zone. UI Map's Schema:

```
<schema>
   <uiMap fkRef="F1-UISRC"/> -
</schema>
```

UI Map's HTML

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
   <tr>
      <td>UI Map with Search </td>
      <td><input oraField="uiMap"></td>
   </tr>
</table>
</body>
<xml>
<root>
   <uiMap/>
</root>
</xml>
</html>
```

In all cases, the HTML rendered is the same.

Initializing Search Fields

This optional attribute is used to initialize search zone filters. Multiple filters may be initialized. This attribute can only be used in conjunction with the oraSearch attribute.

Syntax	Valid Values	Field Value Options	Comments
oraSearchField=" "	One or more pairs of field name and field value separated by colon. Each pair is separated by a semi-colon. **oraSearchField="fieldName:fieldValue: ..."** The field name is used to identify the zone filter to initialize when the search is launched. The field name must match the value of the searchField mnemonic specified on a search zone user filter or hidden filter parameter.	No value	If you do not specify a field value, then the value of the input element containing the oraSearchField attribute will be used.
		XPath	Indicate the XPath to the schema element that contains the value to use.
		'literal'	Indicate a literal value to supply.

Note: If you do not specify an oraSearchField attribute and the schema element has a search enabled fkRef specified, the framework automatically builds an oraSearchField where the field name is equal to the FK reference's key (MD) fields.

Warning: The pop-up explorer zone can be invoked one of two ways: By clicking on the search button, or by hitting the enter key from the field to the left of the button. If you click on the button then no search field information will be passed to the zone. Search field information is only used to initialize zone filter values when enter is pressed.

Copyright © 2024, Oracle and/or its affiliates.

Two filter values are initialized as shown in the following example:

```
<schema>
   <bo/> -
   <uiMap/>
</schema>
```

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
   <tr>
      <td>UI Map with Search </td>
      <td><input oraField="uiMap" oraSearch="F1-UISRCH" oraSearchField="MAP_CD; -
BUS_OBJ_CD:bo;"></td>
    </tr>
</table>
</body>
<xml>
<root>
   <bo/>
   <uiMap/>
</root>
</xml>
</html>
```

Mapping Returned Search Fields

This optional attribute is used to direct values returned by the search zone. Multiple fields may be specified. This attribute can only be used in conjunction with the oraSearch attribute.

Syntax	Valid Values	Field Value Options	Comments
oraSearchOut=" "	One or more pairs of field name and field value separated by colon. Each pair is separated by a semi-colon. **oraSearchOut="field name:xpath target; ..."** The field name is used to identify the search result returned from the query zone. The field name must match the ELEMENT_NAME mnemonic defined within the explorer zone's search results parameter.	No value	If you do not specify a field value, then the input element containing the oraSearchField attribute receives the returned value.
		XPath	Indicate the XPath to the schema element that should receive the returned value.

Note: If you do not specify an oraSearchOut attribute, the framework will build a default where the field name will be set equal to the oraSearchField's field name.

Two values are returned in the following example:

```
<schema>
   <bo/>
   <mo/>
</schema>
```

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
   <tr>
      <td>BO Search </td>
```

Copyright © 2024, Oracle and/or its affiliates.

```
      <td><input oraField="bo" oraSearch="Z1-BOSRCH" oraSearchOut="BUS_OBJ_CD; -
MO_CD:mo;"></td>
   </tr>
</table>
</body>
<xml>
<root>
   <bo/>
   <mo/>
</root>
</xml>
</html>
```

Display Errors

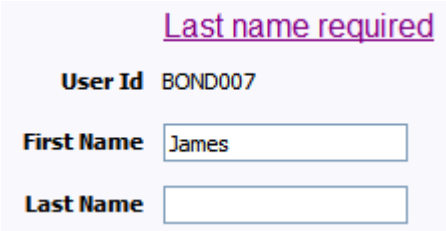
Display Error Variables

One of the following error variables may be displayed.

Syntax
oraErrorVar="ERRMSG-TEXT"
oraErrorVar="ERRMSG-LONG"
oraErrorVar="ERRMSG-CATEGORY"
oraErrorVar="ERRMSG-NUMBER"

```
...
<table width="100%" cellpadding="12">
   <tr class="oraErrorText">
      <td>
         <a href="" onclick="oraShowErrorAlert(); return false;">
            <span  class="oraErrorText" oraErrorVar="ERRMSG-TEXT"></span>
         </a>
      </td>
   </tr>
</table>
...
```

HTML rendered



Highlight a Field in Error

Note: For more information on throwing an error, refer to the Terminate Statement in the Edit Data Syntax.

Syntax	Comments
oraError="automate:true|false; prefix: "	Specifying **automate:true** automatically enables highlighting of the element in error when issuing an error. Note that **true** is the default and doesn't need to be specified. Specify **automate:false** to turn off field highlighting.
	The system uses a match on the element name referenced in the error to the element names in the UI map. If the elements in the schema are within an XPath

Copyright © 2024, Oracle and/or its affiliates.

Syntax	Comments
	that may not match what is referenced by the error, use **prefix:XPath** to specify that.

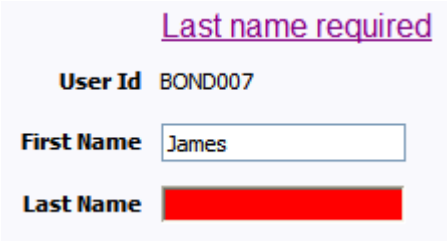
Note: A pair of stylesheet references, cisEnabled and cisDisabled, must be specified for reference of the oraError style. The stylesheet controls how the field in error will be rendered. If you want to alter the rendering you must override the oraError style.

The following HTML example shows that the elements in the map are defined within a group called **boGroup**. The element name returned by the error will not include this group so in order for the field highlighting to work properly the **prefix:** attribute must indicate the group name.

```
<html>
<head>
   <title>User Zone Input</title>
   <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
   <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body oraError="automate:true; prefix:boGroup">
<table width="100%" cellpadding="12">
   <tr class="oraErrorText">
      <td>
         <a href="" onclick="oraShowErrorAlert(); return false;">
            <span  class="oraErrorText" oraErrorVar="ERRMSG-TEXT"></span>
         </a>
      </td>
   </tr>
</table>
<table width="100%" border="0" cellpadding="4">
   <tr style="padding-top:30px;">
      <td align="right" class="label">User Id</td>
      <td>
         <span oraField="boGroup/userId" class="normal"/>
      </td>
   </tr>
   <tr>
      <td align="right" class="label">First Name</td>
      <td>
         <input oraField="boGroup/firstName" class="normal"/>
      </td>
   </tr>
   <tr>
      <td align="right" class="label">Last Name</td>
      <td>
         <input oraField="boGroup/lastName" class="normal"/>
      </td>
   </tr>
</table>
</body>
<xml>
<root>
   <boGroup>
      <userId>BOND007</userId>
      <firstName>James</firstName>
      <lastName></lastName>
   </boGroup>
</root>
</xml>
</html>
```

HTML rendered, where the error element thrown is equal to 'lastName':

Copyright © 2024, Oracle and/or its affiliates.



Overriding the Error Element Name

In the rare occasion where the element name returned by the error doesn't match the element name in the map, you can add an explicit attribute to indicate the error element name.

Syntax	Valid Values	Comments
oraErrorElement=	"element name"	The element name referenced here must exactly match the name of the error element assigned when the error was thrown. More than one HTML field can be referenced by the same error element name.

Note: A pair of stylesheet references, cisEnabled and cisDisabled, must be specified for reference of the oraError style. The stylesheet controls how the field in error will be rendered. If you want to alter the rendering you must override the oraError style.

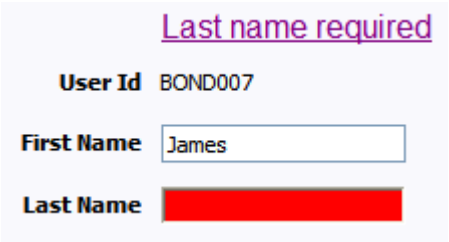
This illustrates a scenario where the element name associated with the error differs from the element name on the map.

```
<html>
<head>
   <title>User Zone Input</title>
   <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
   <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body>
<table width="100%" cellpadding="12">
   <tr class="oraErrorText">
      <td>
         <a href="" onclick="oraShowErrorAlert(); return false;">
            <span  class="oraErrorText" oraErrorVar="ERRMSG-TEXT"></span>
         </a>
      </td>
   </tr>
</table>
<table width="100%" border="0" cellpadding="4">
   <tr style="padding-top:30px;">
      <td align="right" class="label">User Id</td>
      <td>
         <span oraField="userId" class="normal"/>
      </td>
   </tr>
   <tr>
      <td align="right" class="label">First Name</td>
      <td>
         <input oraField="firstName" class="normal" oraErrorElement="firstName"/>
      </td>
   </tr>
   <tr>
      <td align="right" class="label">Last Name</td>
      <td>
         <input oraField="familyName" class="normal"  oraErrorElement="lastName"/>
      </td>
   </tr>
</table>
</body>
<xml>
```

Copyright © 2024, Oracle and/or its affiliates.

```
<root>
   <userId>BOND007</userId>
   <firstName>James</firstName>
   <familyName></familyName>
</root>
</xml>
</html>
```

HTML rendered.



Display Error Pop-Up

When the error text is displayed, this function may be used to pop-up the standard error dialog (which displays more information) when a user clicks the error message.

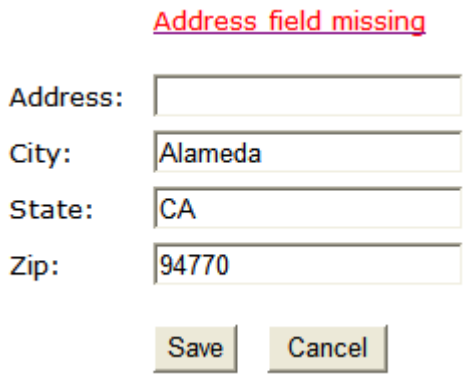
Syntax
oraShowErrorAlert(); return false;

```
<html>
<head>
   <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
   <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body>
<table width="100%" cellpadding="12">
   <tr class="oraErrorText">
      <td>
         <a href="" onclick="oraShowErrorAlert(); return false;">
            <span  class="oraErrorText" oraErrorVar="ERRMSG-TEXT"></span>
         </a>
      </td>
   </tr>
</table>
<table>
   <tr>
      <td >Address:</td>
      <td><input type="text" oraField="address"/></td>
   </tr>
   <tr>
      <td>City:</td>
      <td><input type="text" oraField="city"/></td>
   </tr>
   <tr>
      <td>State:</td>
      <td><input type="text" oraField="state"/></td>
   </tr>
   <tr>
      <td>Zip:</td>
      <td><input type="text" oraField="zip"/></td>
   </tr>
   <tr>
      <td/>
      <td style="padding-top=15px;">
         <oraInclude map="F1-SaveCancelButtons"/>
      </td>
   </tr>
</table>
```

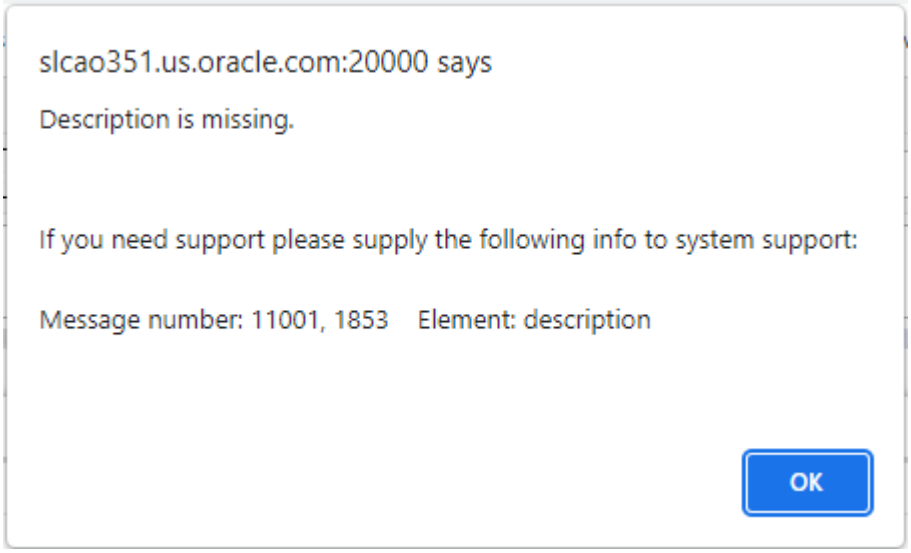
Copyright © 2024, Oracle and/or its affiliates.

```
</body>
<xml>
<root>
   <address>123 Main St</address>
   <city>Alameda</city>
   <state>CA</state>
   <zip>94770</zip>
</root>
</xml>
</html>
```

HTML rendered.



Standard error pop-up dialog launched via click on error message:



Fire JavaScript for Browser Events

Working with the JavaScript Framework

There are many JavaScript events that can be used within the HTML/Javascript environment. These include events such as onLoad, onBlur, onChange, etc. The UI Map Framework also makes use of some of these events. It is important that any UI Map you develop works with the framework in order to obtain consistent results (events may not always be executed in the same order at all times!).

Copyright © 2024, Oracle and/or its affiliates.

Warning:

The following describes the recommended approach for safely handling loading and processing field updates in your UI Maps.

If JavaScript is required within an XHTML UI Map or fragment, it will be necessary to bound it within a `![CDATA[ ]]` tag to ensure a valid XML document. Note that the tags themselves may need to be commented out to promote compatibility with older browsers. For example:

```
<script type="text/javascript">
/* <![CDATA[ */

//
// javascript
//

/* -]]> */

</script>
```

Element Change Event

Syntax	Valid Values
oraChange=" "	A JavaScript function.

At the time of UI Map load, if there is an event handler already attached to an HTML element, the framework removes it and attaches a combined event handler. The combined handler calls any **framework handler first** and then calls the other (custom) handlers.

Warning: Note that the function must not be used to execute logic that will modify the associated field data value again - or an endless loop will occur.

In the following example the **oraInvokeBS** function is executed when the button is clicked.

```
<html>
  <head>
    <title>oraInvokeBS test</title>
  </head>
  <body>
    <table>
      <tr>
        <td>User Id:</td>
        <td>
          <input oraField= -"xmlGroup/userId"/>
          <input type="button" value="Search" -
oraChange="oraInvokeBS('UserSearch','xmlGroup');"/>
        </td>
      </tr>
      <tr>
        <td/>
        <td>Search Results</td>
      </tr>
      <tr>
        <td/>
        <td>
          <table oraList="xmlGroup/searchList">
            <tr>
              <td><span oraField="userId"></span>
              </td>
            </tr>
          </table>
        </td>
      </tr>
    </table>
  </body>
  <xml>
    <root>
      <xmlGroup>
        <userId/>
        <searchList>
            <userId></userId>
```

Copyright © 2024, Oracle and/or its affiliates.

Hide Elements

Hide Using a Function

The system provides multiple techniques for hiding an HTML element.

Syntax	Valid Values	Comments
oraDisplayNone(item);	(item, 'XPath', 'value', 'operator')	Used to hide an element based on the value of another element (referenced using its XPath). Enter a value of '' to interrogate a blank value. By default the operator is **'='**. This may be set instead to another operator such as **'!='**, **'>'**, or **'<'**.
	(item, function name,**true** | **false**)	Used to indicate a JavaScript function, which must return a Boolean.
oraIsSuppress="true"		Use this option for fields that need to be hidden always and not dependent on something else. Set this attribute on the HTML element (div element).

- Example where the User Id label is hidden when no User Id value exists.

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
    <tr>
        <td oraLoad="oraDisplayNone(item,'userId','')">User Id: </td>
        <td><span oraField="userId"></span></td>
    </tr>
</table>
</body>
<xml>
<root>
    <userId></userId>
</root>
</xml>
</html>
```

- Example where the Save button is hidden when the user does not have security access to the action of change ('C') for the application service 'F1-DFLTS'.

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
    <tr>
   <td oraLoad="oraDisplayNone(item, oraHasSecurity('F1-DFLTS', -'C'), false -);">
      <input name="Save" type="button" onclick="oraInvokeBO('CM-
IndividualTaxpayer', null, -'replace')"/>
   </td>
    </tr>
</table>
</body>
<xml>
<root>
    <userId></userId>
</root>
</xml>
</html>
```

Check User's Security Access

The system provides two functions to check a user's security access to a given application service and access mode. These are commonly used for hiding elements.

Copyright © 2024, Oracle and/or its affiliates.

Syntax	Parameters
oraHasSecurity( )	'Application Service code'
	'Access Mode'
oraHasSecurityPath('x','y')	'Application Service XPath'
	'Access Mode XPath'

See the previous section for an example of the **oraHasSecurity** function. The following shows an example where the status button is hidden when the user does not have security access to the access mode 'ACT' of the application service 'FORMTST'.

```
<html>
<body>
<table>
    <tr>
   <td oraLoad="oraDisplayNone(item, -
oraHasSecurityPath('appService', -'accessMode'), false -);">
      <input oraField="statusLabel" type="button" -
onclick="oraRunScript('UpdateState','status');"/>
   </td>
    </tr>
</table>
</body>
<xml>
<root>
    <status>ACTIVATE</status>
<statusLabel>Activate</statusLabel>
<appService>FORMTST</appService>
    <accessMode>ACT</accessMode>
</root>
</xml>
</html>
```

Invoke Schema Based Services

The system provides functions for invoking a business object, business service or service script.

Invoke BO Function

This function is used to perform a BO interaction directly from the UI map's HTML. It returns a 'true' or a 'false' depending on whether the invocation encounters an error.

Syntax	Parameters	Comments
oraInvokeBO( )	'BO Name'	
	'XPath' or **null**	Identifies a group element via XPath. If you specify the word **null**, then the entire embedded XML object will be passed.
	'action'	Indicate the action to use. Valid values are **add**, **delete**, **read**, **update**, **replace**, **fastAdd** and **fastUpdate**.
		Fastpath: Refer to BO Actions for more information about the various BO actions.

Example with the statement invoked in a JavaScript function.

```
function invokeBO {
     if (!oraInvokeBO('F1-User','xmlGroup','read')) {
```

Copyright © 2024, Oracle and/or its affiliates.

```
          oraShowErrorAlert();
          return;
}
}
```

Example with the statement invoked within onClick.

```
<input type="button"  class="oraButton" oraMdLabel="DISPLAY_LBL" -
onClick="oraInvokeBO('F1-User');"/>
```

Invoke BS Function

This function is used to perform a business service interaction directly from the UI map's HTML. It returns a 'true' or a 'false' depending on whether the invocation encounters an error.

Syntax	Parameters	Comments
oraInvokeBS( )	'BS Name'	
	'XPath' or **null**	Identifies a group element via XPath. If you specify the word **null**, then the entire embedded XML object will be passed.

Example with the statement coded within a JavaScript function.

```
function invokeBS {
     if (!oraInvokeBS('F1-UserSearch','xmlGroup')) {
          oraShowErrorAlert();
          return;
  -}
}
```

Example with the statement invoked via onClick.

```
<input type="button"  class="oraButton" oraMdLabel="DISPLAY_LBL" -
onClick="oraInvokeBS('F1-RetrieveInfo');"/>
```

Invoke SS Function

This function is used to perform a service script interaction directly from the UI map's HTML. It returns a 'true' or a 'false' depending on whether the invocation encounters an error.

Syntax	Parameters	Comments
oraInvokeSS( )	'Service Script Name'	
	'XPath' or **null**	Identifies a group element via XPath. If you specify the word **null**, then the document belonging to the parent node will be passed. If the parent node is not enough, then the entire document can always be passed using the following syntax: `oraInvokeSS('service script', - null, null, [$SEQUENCEID])`

Example with the statement invoked within a JavaScript function:

```
function invokeSS {
     if (!oraInvokeSS('F1-GetUsers','xmlGroup')) {
          oraShowErrorAlert();
```

Copyright © 2024, Oracle and/or its affiliates.

```
            return;
}
}
```

Example with the statement invoked within onClick.

```
<input type="button"  class="oraButton" oraMdLabel="DISPLAY_LBL" -
onClick="oraInvokeSS('F1-GetUserInfo');"/>
```

Refresh a Rendered Map or Portal Page

Refresh Map

This function is used to 'Refresh' only the map zone issuing the command.

Syntax
oraRefreshMap;

```
...
    <tr>
        <td/>
        <td><input type="button" onclick="oraRefreshMap();" value="Refresh"/></td>
    </tr>
...
```

Refresh Page

This function is used to refresh all zones in the portal.

Syntax
oraRefreshPage;

```
...
    <tr>
        <td/>
        <td><input type="button" onclick="oraRefreshPage();" -
value="Refresh"/></td>
    </tr>
...
```

Embed Framework Navigation

Navigate using Navigation Option

This function is used to navigate to another page using the information defined on a navigation option.

Syntax	**Parameters**
oraNavigate( );	'Navigation Option code'
	'Key XPath'

Warning: This function is only intended for a UI map defined within a portal zone. It should not be used within a UI map launched by a BPA script.

The following example exhibits two possible uses of this function: as a URL and as a button. Note that the UI Map schema must contain a fkRef attribute. Refer to FK Reference Formatting for more information.

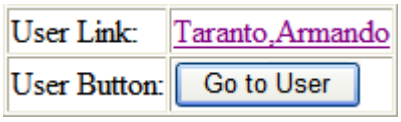
```
<schema>
```

Copyright © 2024, Oracle and/or its affiliates.

```
    <userId fkRef="CI_USER"/>
</schema>
```

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
    <tr>
       <td>User Link: </td>
       <td><a href="" onclick="oraNavigate('userMaint','userId'); return false;">
         <span oraField="userId" oraType="fkRef:CI_USER"></span></a>
       </td>
    </tr>
    <tr>
        <td>User Button: </td>
        <td><input type="submit" onclick="oraNavigate('userMaint','userId')" -
          value="Go to User"/></td>
    </tr>
</table>
</body>
<xml>
<root>
    <userId>SPLAXT</userId>
</root>
</xml>
</html>
```

HTML rendered.



Launch BPA Script

Launch BPA Script

Syntax	Parameters	Comments
oraRunScript( );	'BPA script code'.	
	'XPath Element'	One or more element values may be passed into the BPA where it may be referenced as temporary variables.

Warning: This function is only applicable to UI maps displayed in portal zones. UI maps launched within a running BPA script cannot directly launch another BPA script from the UI map's HTML. Instead, return a value from the UI map and execute a Perform Script or Transfer Control step type.

Note: It is incumbent on the script author to pull information out of temporary storage in the initial steps of the script.

In the following example, a temporary variable named 'personId' is created with value '1234567890' and the BPA script named 'Edit Address' is launched.

```
<html>
<body>
<table>
   <tr>
      <td>
         <div oraField="address"></div>
         <span oraField="city"></span>
         <span>,</span>
         <span oraField="state"></span>
         <span oraField="zip"></span>
```

Copyright © 2024, Oracle and/or its affiliates.

```
         <span oraField="country"></span>
         <a href="" onClick="oraRunScript('Edit Address','personId');">edit</a>
      </td>
   </tr>
</table>
</body>
<xml>
<root>
   <personId>1234567890</personId>
   <address>123 Main St</address>
   <city>Alameda</city>
   <state>CA</state>
   <zip>94770</zip>
</root>
</xml>
</html>
```

HTML rendered.

123 Main St
Alameda, CA 94770 edit

Launch BPA Script With Values

This function is used to launch a BPA, providing name/value pairs to push into temporary storage. Multiple values can be passed. The BPA script can then reference the temporary variables by name.

Syntax	Parameters	Comments
oraRunScriptWithValues( );	'BPA script code'.	
	'XPath Element Name':value	One or more pairs of element names and values.

Note: You would use this JavaScript function, instead of oraRunScript, if you need to push values to the BPA script that are not located in the UI Map's XML structure.

In the example below, a JavaScript function named 'editUser()' is responsible for launching the BPA script named 'UserEdit'. The temporary variable named 'userId' will be created with value 'CMURRAY'.

```
<html>
<head>
<script type="text/javascript">
 -
function editUser() {
    var values = {'userId': -'CMURRAY'};
    oraRunScriptWithValues('UserEdit', values);
    return;
}
 -
</script>
</head>
<body>
...
</body>
</html>
```

Exit UI Map with Bound Values

This function is used to exit a UI Map. When you quit the map you can specify a value to return to the script and, in addition, whether to return updated XML.

Copyright © 2024, Oracle and/or its affiliates.

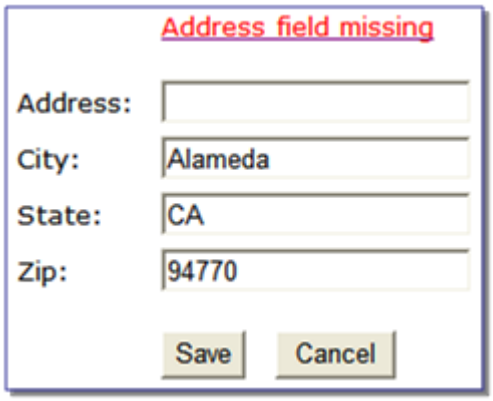
Syntax	Parameters	Comments
oraSubmitMap( );	'Return Value'	
	Boolean value	Indicates if the updated XML should be returned. Default is **true**.

In the following example, the Save button will return updated information, the Cancel button will not.

```
<html>
<body>
<table>
   <tr>
      <td/>
      <td style="padding-bottom:15px;">
         <a href="" onclick="oraShowErrorAlert(); return false;">
         <span oraErrorVar="ERRMSG-TEXT"></span></a>
      </td>
   </tr>
   <tr>
      <td >Address:</td>
      <td><input type="text" oraField="address"/></td>
   </tr>
   <tr>
      <td>City:</td>
      <td><input type="text" oraField="city"/></td>
   </tr>
   <tr>
      <td>State:</td>
      <td><input type="text" oraField="state"/></td>
   </tr>
   <tr>
      <td>Zip:</td>
      <td><input type="text" oraField="zip"/></td>
   </tr>
   <tr>
      <td/>
      <td style="padding-top=15px;">
         <input type="button" value="Save" onClick="oraSubmitMap('SAVE');"/>
         <input type="button" value="Cancel" onClick="oraSubmitMap('CANCEL',false);"/
>
      </td>
   </tr>
</table>
</body>
<xml>
<root>
   <address>123 Main St</address>
   <city>Alameda</city>
   <state>CA</state>
   <zip>94770</zip>
</root>
</xml>
</html>
```

Save and Cancel buttons rendered:

Copyright © 2024, Oracle and/or its affiliates.



Include a Map Fragment

This function is used to embed a map fragment within another UI map. Note that it is possible to use the include node within a map or a map fragment.

Syntax	Parameters	Comments
<oraInclude map=' ' prefixPath=' '/>	map='UI Map Code'	
	prefixPath='Xpath'	Optionally specify an xpath prefix to be appended onto every included oraField, oraLabel, oraList, oraSelect valuePath and descPath, oraDownloadData, and oraUploadData attribute value defined within the included UI map fragment's HTML.
		Note: This functionality only applies to XPath attribute values when those values do not appear beneath an oraList attribute. Any XPath value within a table containing an oraList attribute will not be affected by a prefixPath.

- An example of a map fragment with two buttons, named 'F1-SaveCancelButtons'.

```
<input onClick ="oraSubmitMap('SAVE');" oraMdLabel="SAVE_BTN_LBL" -
class="oraButton" -
type="button"/>
<input onClick ="oraSubmitMap('CANCEL',false);" oraMdLabel="CANCEL_LBL" -
class="oraButton" -
type="button"/>
```

- An example of a map that includes the map fragment named 'F1-SaveCancelButtons'.

```
...
<tr>
   <td colspan="2" align="center">
<oraInclude map="F1-SaveCancelButtons"/>
   </td>
</tr>
...
```

Include a Web Component

A web component is a custom HTML element that produces a specific user interface feature. Each supported web component typically has one or more defined HTML attributes that affect how the web component functions and how it appears. The more complex web components may have a corresponding system configuration object that defines the behavior and information displayed by the resulting user interface element. For example the web component

Copyright © 2024, Oracle and/or its affiliates.

that produces a "tree" of information has a Tree configuration table, which defines the details about how to retrieve information for the tree and how to display each node.

Common Attributes

Some common attributes that may be used for each web component are:

- A reference to the specific instance of the configuration object that governs the details to produce.
- The ability to pass in context data as name and value pairs.
- A 'mode' that support showing a preview of the feature

Some attributes support a **single value** that is referenced. For these types of attributes, the value may be provided using one of the following references:

- Literal value. The value can be referenced directly. Example:

```
treename="F1-Tree"
```

- XPath Reference. XPath references must use the explicit syntax **x[value]**. Example:

```
treename="x[migrationTree]"
```

- Global or Portal context reference. These references must use the explicit syntax **c[value]**.

```
treename="c[TREE_NAME]"
```

- Field label reference. These references must use the explicit syntax **md[value]** .

```
text=md[F1_ADD_LBL]"
```

- Field help text reference. These references must use the explicit syntax **mdh[value]** .

```
text=mdh[F1_OPERATIONS_LBL]"
```

For **complex attributes** that support one or more name / value pairs, the syntax for defining the names and values follows the same rules as above for literal values, XPath references and context. Colons (":") should be used to separate the name from the value and semi-colons (";") separate each pair.

```
context="FIELD_NAME1:x[value1];FIELD_NAME2:F1AB;x[fieldName3]:c[FIELD_NAME3];"
```

A given web component may have additional HTML attributes specific to its functionality.

Note: Any UI map that defines a web component must include the F1-OJETLIBSR map fragment within the "body" tag.

```
<body>
<oraInclude map="F1-OJETLIBSR"/>
..
</body>
```

Note: Standard HTML attributes may also be provided (e.g ID=".." and style=".."). These are defined as normal. It should be noted that the web components themselves may control some styling aspects directly and thus override any style settings provided on the HTML web component. For example, the Contextual Insights configuration provides settings for text color that would override any color defined in the HTML.

The following sections describe each supported web component.

Trees

Refer to Trees for more information about tree configuration.

Copyright © 2024, Oracle and/or its affiliates.

Syntax	Parameters	Comments
<ou-tree...></ou-tree>	**treename="singleValue"**	You must supply a reference to the name of the tree object that defines the configuration of the tree to display. This attribute expects a single value that follows the syntax described in the single value section above.
	context="..."	Optionally pass in one or more name / value pairs. The syntax follows the example shown in the complex attributes information above.

The following example uses an explicit reference to a tree and does not pass any context. The particular tree relies on global context, which the tree node algorithm has access to.

```
<ou-tree treename="C1-PremiseTree"></ou-tree>
```

Contextual Insight

Refer to Contextual Insights for information about the configuration related to defining contextual insights.

Syntax	Parameters	Comments
<ou-insights ...></ou-insights>	**insightType="singleValue"**	Use this attribute to specify an insight type to display. Either this attribute or the insight class attribute must be supplied. This attribute expects a single value that follows the syntax described in the single value section above.
	insightClass="singleValue"	Use an insight class to display all the insight types for the insight group related to this class. This attribute expects a single value that follows the syntax described in the single value section above.
	context="..."	Optionally pass in one or more name / value pairs. The syntax follows the example shown in the complex attributes information above.
	mode="preview"	Optionally pass in a mode of **preview** to indicate to the underlying insight algorithm that prepares the data that the insight is being called in preview mode, with no context available. Ideally the algorithm returns sample information.
	class=" "	Optionally provide a CSS helper class to control aspects of the rendered insight. Supported values: **class="card-full-width"** - Applicable for card insights. Use this option class to ensure that the width for a card insight adjusts to the width of the zone that contains it. Card insights that don't set this parameter use a fixed width.

Copyright © 2024, Oracle and/or its affiliates.

The following example is from the insight preview map where the insight type is the one in context. No context data is needed and the mode is 'preview'.

```
<ou-insights insightType="x[insightType]" mode="preview"></ou-insights>
```

The following example uses an insight class and passes in context values from XPath elements with explicitly defined context names.

```
<ou-insights insightClass="C1CI" -
context="PER_ID:x[personId];ACCT_ID:x[accountId];PREM_ID:x[premiseId];"></ou-
insights>
```

Button

The button web component provides support for the standard set of button formats used within the system.

Syntax	Parameters	Comments
<ou-button...></ou-button>	**type=”singleValue”**	You must define the type of button to display. This attribute expects a single value that follows the syntax described in the single value section above. The valid type values are: • **image** - image only • **text** - text only • **imageText** - image followed by text • **menu** - text plus a dropdown caret to open the menu • **imageMenu** - image plus a dropdown caret to open the menu If no type attribute is supplied, a ‘broken button’ image will be displayed.
	img=”singleValue”	Define the path of the image to display. Either this attribute or the image reference attribute must be supplied for a button type that includes an image. This attribute expects a single value that follows the syntax described in the single value section above. Note that for SVG images, if the reference includes only the SVG file name with no path, the path is assumed to be the Framework SVG repository.
	imgRef=”singleValue”	Define the display icon reference of the image to be used. Either this attribute or the image attribute must be supplied for a button type that includes an image. This attribute expects a single value that follows the syntax described in the single value section above.
	text="..."	Define the various text elements needed by the button. The syntax follows the example shown in the complex attributes information above. The valid text element name values are: • **label** - the label text to be used for the button • **tooltip** - the hover text to be used for the button • **aria-label** - the ARIA label for the button

Copyright © 2024, Oracle and/or its affiliates.

Syntax	Parameters	Comments
	menu=”singleValue”	Define the menu to be used with the button. This attribute must be supplied for a button type that includes a menu control. This attribute expects a single value that follows the syntax described in the single value section above.
	disabled=”singleValue”	This is a standard HTML attribute. A value of "true" or just the presence of the text "disabled" will result in an initially disabled button.
	security="..."	Optionally define the security settings for the button. The button will be hidden if the user does not have the appropriate security access. The syntax follows the example shown in the complex attributes information above. The valid security element name values are: • **service** - the application service whose security access governs access to the button • **accessMode** - the specific access mode for the service that determines a more granular level of access to the button
	clickAction="..."	Optionally define the action to be taken when the button is ‘clicked’. The syntax follows the example shown in the complex attributes information above. There are specific ‘name’ values for the attribute as follows: • **action** - the action to be taken. Valid values are ‘BPA’ and ‘NAVIGATE’ • **script** - the script to be performed when the action is ‘BPA’ • **navopt** - the navigation option to be used when the action is ‘NAVIGATE’ The attribute may also include other name/value pairs that define information to be passed into the script or used as navigation context fields. In addition to the security attribute settings, the system will check that the user has access to the script or navigation path before performing the action. The standard HTML ‘onClick’ attribute may still be used. if the HTML includes an ‘onClick’attribute as well as the ‘clickAction’, the ‘onClick’ attribute will take precedence

The following is an example of a simple button with text.

```
<ou-button type="text" text="label:md[F1_BOOKMARK_LBL];"></ou-button>
```

The following is an example of an icon button using a display icon reference.

```
<ou-button type="image" imgRef="F1CMNT"></ou-button>
```

Copyright © 2024, Oracle and/or its affiliates.

The following is an example of an image menu button.

```
<ou-button type="imageMenu" imgRef="F1STTGS" menu="CI_CONTEXTSCRIPT" -
text="tooltip:md[F1INSTY_VIS_LBL];></ou-button>
```

The following is an example of text button that navigates to another page.

```
<ou-button type="text" text="label:md[ALG_CD];tooltip:md[GOTO_ALG]" -
clickAction="action:NAVIGATE; navopt:f1algqTabMenu; ALG_CD:x[algorithm];></ou-button>
```

Show Schema Default on Add

Default values within in the UI map's schema will be displayed on a UI map's input fields if an embedded <action> node has a value of 'ADD' or blank.

Syntax
<action>ADD</action>
<action> </action>

The schema default for the <description> element will be displayed:

```
<schema>
    <action/>
    <boGroup type="group">
        <key/>
        <description default="enter description here"/>
    </boGroup>
</schema>
```

```
<html>
<body>
<table summary="" border="1" cellpadding="1" cellspacing="1">
    <tr>
        <td>Description   </td>
        <td><input oraField="boGroup/description"></td>
    </tr>
</table>
</body>
<xml>
<root>
    <action>ADD</action>
    <boGroup>
        <key/>
        <description/>
    </boGroup>
</root>
</xml>
</html>
```

HTML rendered.

Configure a Chart

In addition to integrating charts with zones, the system supports using the same attributes within the UI Map HTML to configure a graphical representation of an XML list. The designer can control the type, size, position, and contents of the chart using these attributes. For details of the attributes and syntax, refer to Data Explorer Graph Configuration.

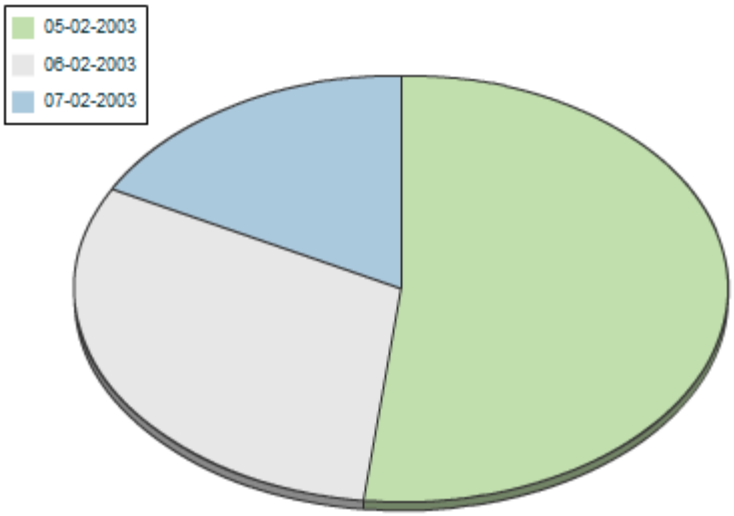
Copyright © 2024, Oracle and/or its affiliates.

Graph Examples

- Sample of a pie chart configuration:

```
<html>
<head>
<title>Pie Chart</title>
</head>
<body>
 -
<div style="width:100%; height:290px;"
    oraChart="type:pie;"
    oraChartSeries1="list:set; labelPath:date; amount:amount; -"
    oraChartBroadcast="BILL_ID:billId;">
</div>
 -
</body>
 -
<xml>
<root>
  <set>
<date>05-02-2003</date>
<amount>163.24</amount>
<billId>592211649441</billId>
  </set>
  <set>
<date>06-02-2003</date>
<amount>97.29</amount>
<billId>592211649442</billId>
  </set>
  <set>
<date>07-02-2003</date>
<amount>54.38</amount>
<billId>592211649443</billId>
  </set>
</root>
</xml>
</html>
```

- A pie chart rendered for a single series:



- Sample of a line, cluster, or stacked graph configuration - each with two series:

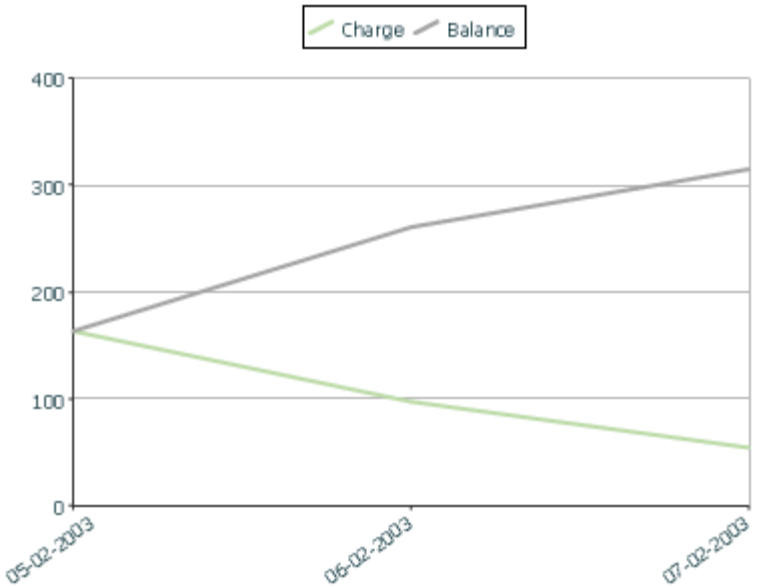
```
<html>
<head>
<title>Stacked Chart</title>
</head>
<body>
```

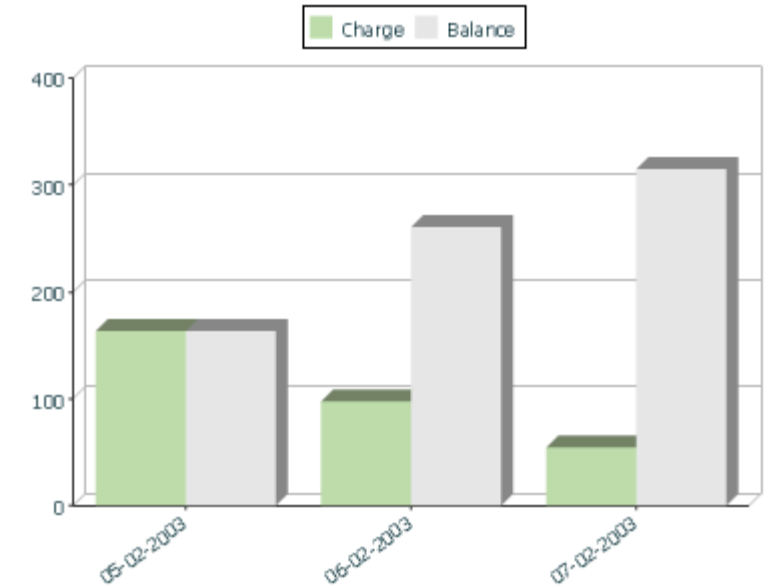
Copyright © 2024, Oracle and/or its affiliates.

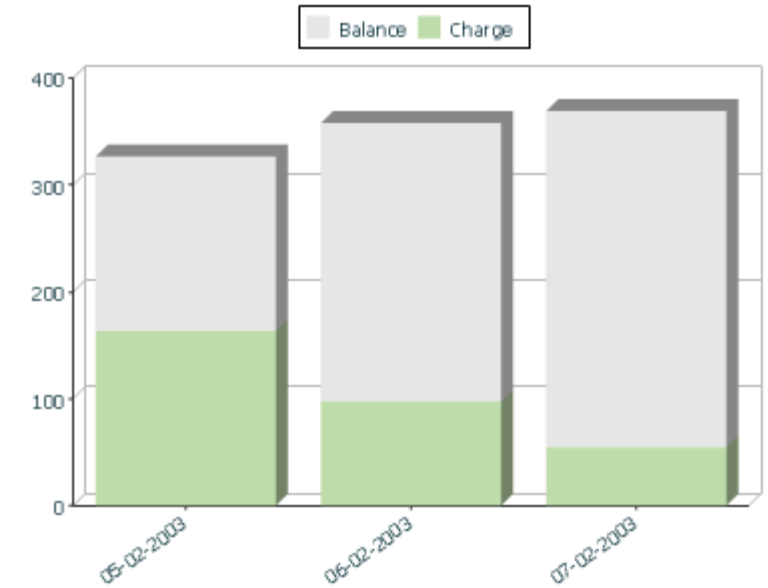
```
 -
<div style="width:100%; height=300px;"
    oraChart="type:line;"
oraChartSeries1="list:set; xaxis:date; label:Charge; amount:amount; -"
oraChartSeries2="list:set; xaxis:date; label:Balance; amount:balance; -"
    oraChartBroadcast="BILL_ID:billId;">
</div>
 -
<div style="width:100%; height=300px;"
oraChart="type:cluster;"
oraChartSeries1="list:set; xaxis:date; label:Charge; amount:amount; -"
oraChartSeries2="list:set; xaxis:date; label:Balance; amount:balance; -"
oraChartBroadcast="BILL_ID:billId;">
</div>
 -
<div style="width:100%; height=300px;"
oraChart="type:stacked;"
oraChartSeries1="list:set; xaxis:date; label:Charge; amount:amount; -"
oraChartSeries2="list:set; xaxis:date; label:Balance; amount:balance; -"
oraChartBroadcast="BILL_ID:billId;">
</div>
 -
</body>
 -
<xml>
<root>
  <set>
<date>05-02-2003</date>
<amount>163.24</amount>
<balance>163.24</balance>
<billId>592211649441</billId>
  </set>
  <set>
<date>06-02-2003</date>
<amount>97.29</amount>
<balance>260.53</balance>
<billId>592211649442</billId>
  </set>
  <set>
<date>07-02-2003</date>
<amount>54.38</amount>
<balance>314.91</balance>
<billId>592211649443</billId>
  </set>
</root>
</xml>
</html>
```

- Three types of chart rendered for two series each: line, cluster, and stacked.

Copyright © 2024, Oracle and/or its affiliates.







Copyright © 2024, Oracle and/or its affiliates.

Upload and Download a CSV File

The following HTML attributes can be used to manage both an upload and a download between a list defined within the map's schema and a CSV (comma separated value) file. Note that this technique is only recommended for a small to medium volume of data, for example no more than several hundred rows. For higher volumes, it is recommended to use batch upload / download functionality instead.

The syntax is **oraUploadData="type:embed;path:list xpath;useLabels:true;showCount:true"**

Upload configuration requires you to name a CSV file to be uploaded, and an XML list as target. By convention, each CSV row will create a separate list instance. Each comma-separated field in the file will be uploaded as a separate element in the list. To embed an upload dialog within a map, the **oraUploadData** attribute must be associated with a container element such as a div, td, or span.

The optional **useLabels:true** value indicates that while parsing the upload CSV file, the headers are expected to be labels

Note: If you do not specify the **useLabels:true** value and the XML target element name is "camelCase" then the corresponding spreadsheet header should be title case with a space between words, e.g.;"Camel Case". Letters and special characters are not considered a different word, for example Address1 must be uploaded into the target XML element address1.

Specifying the optional **showCount:true** value will display the number of records uploaded.

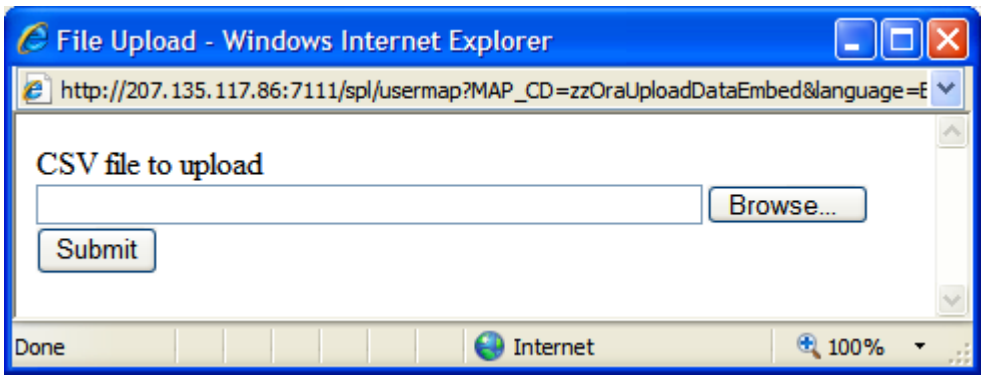
CAUTION: If you are using a grid in conjunction with the **oraUploadData** function, then you must maintain the grid's list with a 'replace' business object action. Refer to BO Replace Action for more information.

Sample of **oraUploadData="embed"** within a div element.

```
<html>
<head>
    <title>File Upload</title>
</head>
<body>
   -
    <div oraUploadData="type:embed;path:myList"> </div>
 -
</body>
 -
<xml>
<root>
    <myList>
<id>838383930</id>
        <name>Janice Smith</name>
    </myList>
    <myList>
<id>737773730</id>
        <name>Bill McCollum</name>
    </myList>
</root>
</xml>
</html>
```

This file upload dialog will be embedded into the body of the page where the oraUploadData is defined.

Copyright © 2024, Oracle and/or its affiliates.



oraUploadData="type:popup;path:list xpath;useLabels:true;showOk:true;showCount:true"

Upload configuration requires you to name a CSV file to be uploaded, and an XML list as target. By convention, each CSV row will create a separate list instance. Each comma-separated field in the file will be uploaded as a separate element in the list. To upload a CSV file using a pop-up dialog, the oraUploadData attribute must be associated with an input element such as a button, text link, or image.

The optional useLabels:true value is used to indicate that while parsing the upload CSV file, the headers are expected to be labels

Note: If you do not specify the useLabels:true value and the XML target element name is "camelCase" then the corresponding spreadsheet header should be title case with a space between words, e.g., "Camel Case". Letters and special characters are not considered a different word, for example Address1 must be uploaded into the target XML element address1.

Specifying the optional showOk:true value will display an "Ok" button once the upload finishes. The popup will stay open until the button is pressed. Additionally, specifying the showCount:true value will display number of records uploaded.

CAUTION: If you are using a grid in conjunction with the **oraUploadData** function, then you must maintain the grid's list with a 'replace' business object action. Refer to BO Replace Action for more information.

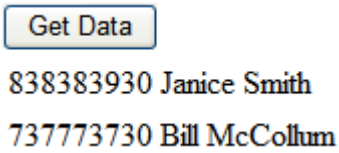
Sample of oraUploadData="popup" associated with a button:

```
<html>
<head>
    <title>File Upload</title>
</head>
<body>
   <input type="button" name="submitButton" oraUploadData="type:popup;path:myList;" -
value='Get Data'>
   <table oraList="myList">
      <tr/>
      <tr>
         <td><span oraField="id"/></td>
         <td><span oraField="name"/></td>
      </tr>
   </table>
</body>
<xml>
<root>
   <myList>
      <id>838383930</id>
      <name>Janice Smith</name>
   </myList>
   <myList>
      <id>737773730</id>
      <name>Bill McCollum</name>
   </myList>
</root>
</xml>
```

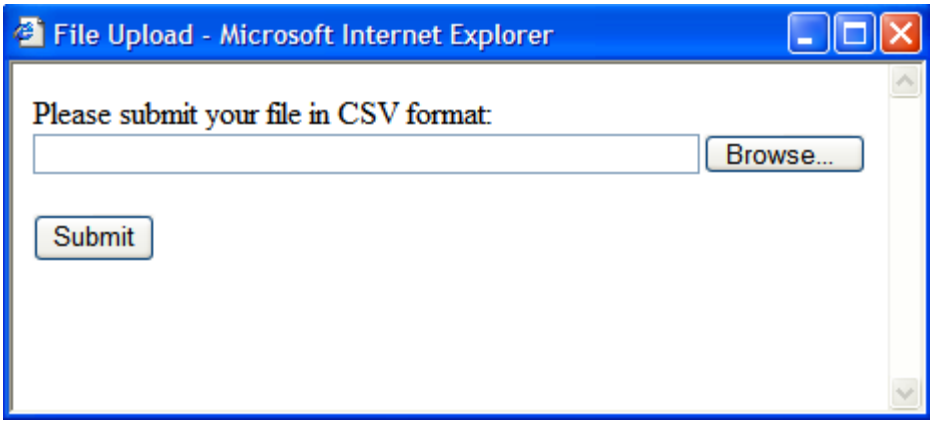
Copyright © 2024, Oracle and/or its affiliates.

```
</html>
```

HTML Rendered:



Pressing the "Get Data" button will launch a standard file upload dialogue (provided by Framework) as shown below.



oraDownloadData="list xpath"

Download configuration requires you to name an XML list to be downloaded. By convention, each list instance will represent a separate row in the created file. By default every element of the list will be comma separated in the file.

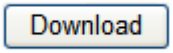
Note: The number formatting is based on the user profile setting. For localities where the decimal symbol is a comma, an implementation may configure a property setting (spl.csv.delimiter.useFromDisplayProfile=true) to cause the system to use a semicolon as the delimiter that separates the elements rather than a comma.

Sample of oraDownloadData.

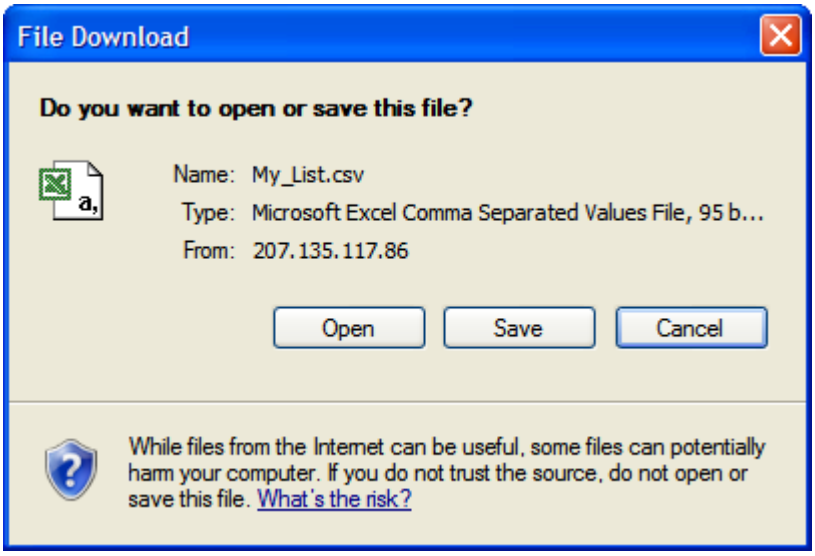
```
<html>
<head>
<title>File Download</title></head>
<body>
<input type="button" name="downloadButton" oraDownloadData="myList" value="Download"/
>
</body>
<xml>
<root>
   <myList>
      <id>881-990987</id>
      <name>John Mayweather</name>
   </myList>
   <myList>
      <id>229-765467</id>
      <name>Anna Mayweather</name>
   </myList>
   <myList>
      <id>943-890432</id>
      <name>Andrew Brewster</name>
   </myList>
</root>
</xml>
</html>
```

HTML Rendered:

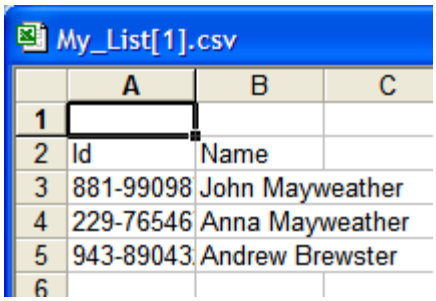
Copyright © 2024, Oracle and/or its affiliates.



Pressing the "Download" button will launch a standard file download dialogue (provided by Framework) as shown below.



A successful download will result in a CSV file:



To download data from a sub list use the attribute oraDownloadDataInList instead of oraDownloadData. The attribute oraDownloadDataInList will have the sub list name. The XPath of the sub list is used to pick data of the specific row from the parent list. Thus only the specific sub list is downloaded.

oraDownloadDataUseLabels="true"

The oraDownloadDataUseLabels attribute can be used in conjunction with the oraDownloadData attribute described above. Specify oraDownloadDataUseLabels if you want the generated CSV file to use the element labels for columns headers rather than element names.

Construct Portal Zone Map Fragments

Portal zones can reference a UI map for the zone header and filter area. This UI map is not a complete HTML document, but is instead configured as a UI Map fragment. When constructing a zone map fragment you can reference the following substitution variables. Note that these variables will be dynamically populated at run time with information particular to the map's zone within the portal:

Variable	Replacement Logic
[$ZONEDESCRIPTION]	Zone's description text.
[$SEQUENCEID]	Zone's sequence ID.

Copyright © 2024, Oracle and/or its affiliates.

Variable	Replacement Logic
[$ZONENAME]	Zone's name.
[$HELPTEXT]	Zone's help text.
[$ZONEPARAMNAME]	Zone parameter's value (or blank if it has not been specified).

Warning:

- Refer to one of the following maps as examples: F1-UIMapHeader and F1-ExplorerHeader.
- These maps make use of the oraInclude tag to incorporate HTML fragments for the header menu and framework actions. Refer to the zone type parameters for the UI Map fragments you should include in your HTML.
- If you wish to have the "help text" icon appear next to your zone description, you should have id="title_[$SEQUENCEID]" on the <td> that contains your description.
- If it is necessary to encapsulate JavaScript within a UI Map fragment, it will be necessary to bound the JavaScript within a `![CDATA[ ]]` tag to ensure a valid XML document. Note that the tags themselves may need to be commented out to promote compatibility with older browsers. For example:

```
<script type="text/javascript">

/*<![CDATA[ */

//
//javascript
//
/*]]> */
</script>
```

Note: If you wish to preserve the values of a filter input field, within a filter map fragment, for the framework 'Go Back' and 'Go Forward' functionality, you must associate the input field (text box, select, etc.) with a unique HTML id. Input field values associated with a unique id will be captured in the framework's 'memento'. The 'memento' is used to rebuild the input map when the portal zone is navigated to using the 'Go Back' or 'Go Forward' functionality.

Note: Many specialized functions exist to manipulate zone behavior, for example:

- **oraGetZoneSequence(zoneName)**. Uses the zone's code to retrieve its sequence number.
- **oraIsZoneCollapsed(sequenceId)**. Uses the zone's sequence to determine if collapsed.
- **oraHandleCollapse(seq)**. Collapse a zone.
- **oraHandleExpand(seq,refresh)**. Expand and/or refresh a zone.

All of these, and many more functions, are located within the JavaScript library userMapSupport.js described below.

Note: When executing oraLoad within a fragment UI map, and you need to execute a JavaScript function during page load (where the function invokes a business object, business service, or service script) you can use the special syntax "oraLoad[$SEQUENCEID]". Refer to the Load Page Event section for more information.

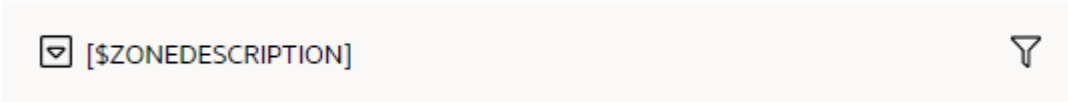
Example of oraLoad[$SEQUENCEID] used within a function:

```
<script type="text/javascript">
function oraLoad[$SEQUENCEID]() {
checkRebateClaimStatus();
}
 -
function checkRebateClaimStatus() {
    var work = id(''analyticsFilterText[$SEQUENCEID]'',
document).cells[0].innerText.split('' -'');
    var rebateClaimId = work[work.length -- 3];
    id(''rebateClaimId'', document).value = rebateClaimId;
oraInvokeSS(''C1-CheckRCSt'',''checkRebateClaimStatus'', false);
```

Copyright © 2024, Oracle and/or its affiliates.

```
    var statusIndicator = id(''statusInd'', document).value;
    if (statusIndicator == -''C1PE'' -|| statusIndicator == -''C1ID'') {
       id(''addRebateClaimLine'', document).style.display = -'''';
    -} else {
id(''addRebateClaimLine'', document).style.display = -''none'';
    -}
}
</script>
```

F1-ExplorerHeader rendered:



Invoking a Business Object

The oraInvokeBO function may be used within a portal zone header or zone filter map. It is similar to the command described in Invoke BO Function which allows for a business object to be invoked within the UI map's HTML. Refer to that section for a description of the first three parameters.

Syntax	Parameters	Comments
oraInvokeBO( )	'BO Name'	
	'XPath' or **null**	
	'action'	
	null	This must be specified as the fourth argument.
	[$SEQUENCEID]	This must be specified as the fifth argument.
	true | false	Specify **true** if the fragment is used within a portal zone header. Specify **false** if the fragment is used with a zone filter map.

Example in a portal zone header:

```
oraInvokeBO('CM-User','xmlGroup','read', null, [$SEQUENCEID], true)
```

Invoking a Business Service

The oraInvokeBS function may be used within a portal zone header or zone filter map. It is similar to the command described in Invoke BS Function which allows for a business service to be invoked within the UI map's HTML. Refer to that section for a description of the first two parameters.

Syntax	Parameters	Comments
oraInvokeBS( )	'BO Name'	
	'XPath' or **null**	
	null	This must be specified as the fourth argument.
	[$SEQUENCEID]	This must be specified as the fifth argument.
	true | false	Specify **true** if the fragment is used within a portal zone header. Specify **false** if the fragment is used with a zone filter map.

Copyright © 2024, Oracle and/or its affiliates.

Example in a portal zone header:

```
oraInvokeBS('CM-UserSearch','xmlGroup', null, [$SEQUENCEID], true)
```

Invoking a Service Script

The oraInvokeSS function may be used within a portal zone header or zone filter map. It is similar to the command described in Invoke SS Function which allows for a service script to be invoked within the UI map's HTML. Refer to that section for a description of the first two parameters.

Syntax	Parameters	Comments
oraInvokeSS( )	'Service Script Name'	
	'XPath' or **null**	
	null	This must be specified as the fourth argument.
	[$SEQUENCEID]	This must be specified as the fifth argument.
	true | false	Specify **true** if the fragment is used within a portal zone header. Specify **false** if the fragment is used with a zone filter map.

Example in a portal zone header:

```
oraInvokeSS('UserSearch','xmlGroup', null, [$SEQUENCEID], true)
```

Detecting Unsaved Changes

Use this function to return a Boolean set to true if there are unsaved changes. The system will interrogate the function when the user attempts to navigate and issue a warning accordingly. This function is only needed if a UI map is using custom javascript to manage elements such that the system is not able to detect whether changes have been made. Also note that it's the responsibility of the UI map javascript to manage the values in the Boolean used for this function.

```
function hasUnsavedChanges(){
       return isDirtyFlag;
   -}
```

Hiding Portal Tabs

The product provides the ability to use JavaScript to hide a tab on the current portal based on some condition using the oraAuthorizeTab JavaScript API. This API accepts a function as a parameter and turns off the tab index indicated.

For example, the UI Map may have a function to turn off one or more tab indexes.:

```
function overrideTabIndex(index){
    if (index == 2) return false;
    if (index == 3) return false;
  -}
```

The JavaScript is referenced "on load":

```
<body class="oraZoneMap"
onLoad="oraAuthorizeTabs(overrideTabIndex);">
```

Required JavaScript Libraries

All of the functionality described in this document depends on a pair of JavaScript libraries. If you are writing and executing your maps entirely within the UI map rendering framework - you do not need to manually insert the following libraries - the framework will insert them for you when the UI Map is rendered.

Copyright © 2024, Oracle and/or its affiliates.

Warning: When executing HTML outside of the framework you must include the following references explicitly within your HTML. In addition, the tool you use to render the HTML must have access to a physical copy of privateUserMapSupport.js for bind support.

src="privateUserMapSupport.js"

Your HTML document must reference this library to execute binding in a stand-alone environment.

Warning: Referencing functions within this JavaScript library is dangerous - because these functions are owned by framework and they may be changed during version upgrade or via the normal patch process.

```
<script type="text/javascript" src="privateUserMapSupport.js"></script>
```

src="userMapSupport.js"

To take advantage of optional toolset features, you must reference this library.

Note: You can reference the functions within this JavaScript library to write custom functions within the UI map..

```
<script type="text/javascript" src="userMapSupport.js"></script>
```

onload="oraInitializeUserMap();"

To execute binding in a stand-alone environment, you must embed the following onload function into the <body> node.

```
<body onload="oraInitializeUserMap();">
```

UI Map Standards

Basic UI Map Templates

All UI Maps share the same basic structure regardless of placement (page area, zone, pop-up) or usage (display only, input).

Sample XML

All information in this document is based upon the following XML structure.

```
<xml>
    <root>
        <address>123 Main St</address>
        <city>Alameda</city>
        <state>CA</state>
        <zip>94770</zip>
        <contactInformation>
            <type>Home Phone</type>
            <number>510-555-2287</number>
        </contactInformation>
        <contactInformation>
            <type>Cell Phone</type>
            <number>510-555-4285</number>
        </contactInformation>
    </root>
</xml>
```

Display Only UI Map

```
<html>
<head>
    <title oraMdLabel="ADDRESS_LBL"></title>
```

Copyright © 2024, Oracle and/or its affiliates.

```
    <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
    <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body class="oraZoneMap">
<table cellspacing="4" width="100%">
    <colgroup>
        <col class="oraLabel oraTableLabel" -/>
        <col class="oraNormal oraTableData" -/>
    </colgroup>
    <tr>
        <td oraLabel="address"></td>
        <td oraField="address"></td>
    </tr>
    <tr>
        <td oraLabel="city"></td>
        <td oraField="city"></td>
    </tr>
    <tr>
        <td oraLabel="state"></td>
        <td oraField="state"></td>
    </tr>
    <tr>
        <td class="oraSectionEnd" oraLabel="zip"></td>
        <td class="oraSectionEnd" oraField="zip"></td>
    </tr>
    <tr>
        <td colspan="2" class="oraSectionHeader" oraMdLabel="CONTACT_LBL"></
td>
    </tr>
    <tr>
        <td colspan="2" class="oraSectionStart oraEmbeddedTable">
            <table oraList="contactInformation" cellspacing="2">
                <thead >
                    <tr>
                        <th class="oraGridColumnHeader" nowrap="nowrap">
                            <span oraLabel="contactInformation/type></span>
                        </th>
                        <th class="oraGridColumnHeader" nowrap="nowrap">
                            <span oraLabel="contactInformation/number"></
span>
                        </th>
                    </tr>
                </thead >
                <tbody>
                    <tr>
                        <td class="oraNormalAlt oraDisplayCell">
                            <span oraField="type"></span>
                        </td>
                        <td class="oraNormal oraDisplayCell">
                            <span oraField="number"></span>
                        </td>
                    </tr>
                </tbody>
            </table>
        </td>
    </tr>
</table>
</body>
<xml>
    <root>
        <address>123 Main St</address>
        <city>Alameda</city>
        <state>CA</state>
        <zip>94770</zip>
```

Copyright © 2024, Oracle and/or its affiliates.

```
        <contactInformation>
            <type>Home Phone</type>
            <number>510-555-2287</number>
        </contactInformation>
        <contactInformation>
            <type>Cell Phone</type>
            <number>510-555-4285</number>
        </contactInformation>
    </root>
</xml>
</html>
```

Input UI Map

```
<html>
<head>
    <title oraMdLabel="ADDRESS_LBL"></title>
    <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
    <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
</head>
<body>
<table width="100%" cellpadding="12">
  <tr class="oraErrorText">
    <td><a href="" onclick="oraShowErrorAlert(); return false;">
      <span  class="oraErrorText" oraErrorVar="ERRMSG-TEXT"></span></a>
    </td>
  </tr>
</table>
<table cellspacing="4" width="100%">
    <colgroup>
        <col class="oraLabel oraTableLabel" -/>
        <col class="oraNormal oraTableData" -/>
    </colgroup>
    <tr>
        <td oraLabel="address"></td>
        <td><input type="text" oraField="address"/></td>
    </tr>
    <tr>
        <td oraLabel="city"></td>
        <td><input type="text" oraField="city"/></td>
    </tr>
    <tr>
        <td oraLabel="state"></td>
        <td><input type="text" oraField="state"/></td>
    </tr>
    <tr>
        <td oraLabel="zip"></td>
        <td><input type="text" oraField="zip"/></td>
    </tr>
    <tr>
        <td colspan="2" class="oraSectionHeader" oraMdLabel="CONTACT_LBL"></
td>
    </tr>
    <tr>
        <td colspan="2" class="oraSectionStart oraEmbeddedTable">
            <table oraList="contactInformation" cellspacing="2">
                <thead >
                    <tr>
                        <th class="oraGridColumnHeaderButton"></th>
                        <th class="oraGridColumnHeaderButton"></th>
                        <th class="oraGridColumnHeader" nowrap="nowrap">
                            <span oraLabel="contactInformation/type></span>
                        </th>
```

Copyright © 2024, Oracle and/or its affiliates.

```
                        <th class="oraGridColumnHeader" nowrap="nowrap">
                            <span oraLabel="contactInformation/number"></
span>
                        </th>
                    </tr>
                </thead >
                <tbody>
                    <tr>
                        <td oraType="addGridRow"></td>
                        <td oraType="deleteGridRow"></td>
                        <td>
                            <input type="text" oraField="type"/>
                        </td>
                        <td>
                            <input type="text" oraField="number"/>
                        </td>
                    </tr>
                </tbody>
            </table>
        </td>
    </tr>
    <tr>
      <td colspan="2" class="oraSectionStart oraEmbeddedTable">
        <table cellspacing="2">
          <tr>
            <td>
              <input class="oraButton" oraMdLabel="C1_SAVE_LBL" -
type="button" -
                onClick="oraSubmitMap('OK');"/>
            </td>
            <td>
              <input class="oraButton" oraMdLabel="CANCEL_LBL" -
type="button" -
               onClick="oraSubmitMap('CANCEL',false);"/>
            </td>
          </tr>
        </table>
      </td>
    </tr>
</table>
</body>
<xml>
    <root>
        <address>123 Main St</address>
        <city>Alameda</city>
        <state>CA</state>
        <zip>94770</zip>
        <contactInformation>
            <type>Home Phone</type>
            <number>510-555-2287</number>
        </contactInformation>
        <contactInformation>
            <type>Cell Phone</type>
            <number>510-555-4285</number>
        </contactInformation>
    </root>
</xml>
</html>
```

Basic HTML and Styles

The basic templates introduced the standard HTML and styles used for UI Maps. These standards are described individually in the following sections.

Copyright © 2024, Oracle and/or its affiliates.

Stylesheets

The styles to apply the standard look to the maps are all contained in stylesheets. These stylesheets should be included in all UI Maps.

```
...
    <link rel="stylesheet" type="text/css" href="cisDisabled.css"/>
    <link rel="stylesheet" type="text/css" href="cisEnabled.css"/>
...
```

Title

Each UI Map should have a <title> tag.

```
...
    <title oraMdLabel="ADDRESS_LBL"></title>
...
```

This will give the UI Map a descriptive title.

- If the UI Map is presented in a "pop-up", the title will be in the window title bar.
- If the UI Map is presented in the page area, the title will be added as a tag to the UI Map and will appear at the top of the UI Map.
- If the UI Map is presented as a zone map, it will be ignored. The <title> tag should still be included in the HTML as standard.

Zone Maps

When the map is presented in a zone as part of a portal, the UI Map should have a border so that the information is "contained" within the zone.

```
...
    <body class="oraZoneMap">
...
```

Page Area Maps vs Pop-Up Maps

The presentation of the UI Maps can vary from design to design. The following standards have been applied to decide when to use a Page Area UI Map and when to use a Pop-Up Map:

- If there are multiple UI Maps in the sequence, always use the Page Area.
- If the UI Map has many input fields, always use a Page Area.
- If the UI Map is a "confirmation" type dialog or only has one or two input fields, use a Pop-Up.

Note: The difference between "just a few input fields" and "many input fields" can be discretionary. The final decision should rest with the dialog designer.

Error Messages

Input maps have a ability to present error messages to the User.

```
...
<table width="100%" cellpadding="12">
  <tr class="oraErrorText">
   <td><a href="" onclick="oraShowErrorAlert(); return false;">
        <span class="oraErrorText" oraErrorVar="ERRMSG-TEXT"></span></a></
td>
  </tr>
</table>
...
```

Copyright © 2024, Oracle and/or its affiliates.

This HTML structure provides the provides the necessary elements and functions to display errors to the User. It should be directly after the <body> tag. When there is no error, nothing will be visible on the UI Map. It will be made visible if an error occurs and the UI Map is re-presented to the User. Clicking on the link (when visible) will result in a pop-up alert appearing with the long error message text.

Standard Layout and Styles

The information is presented on the UI Map by using a <table> to organize the information in rows and columns.

```
...    -
<table cellspacing="4" width="100%">
<colgroup>
<col class="oraLabel oraTableLabel" -/>
    <col class="oraNormal oraTableData" -/>
    </colgroup>
...
```

The <colgroup> and <col> tags allow for the application of classes to the columns (the label is in the first column and the data is in the second column.). Using these tags mean that the class attribute (to apply styles) does not need to be defined on every <td>.

Grids (Tables of Data)

A UI Map could contain information that is best presented as a grid. These are referred to as "Embedded Tables". The embedded table can be used to display information or input information.

Example Embedded Table HTML

The embedded table will be included within a row (<tr>) of the basic layout:

```
...
<tr>
    <td colspan="2" class="oraEmbeddedTable">
        <table oraList="contactInformation" cellspacing="2">
            <thead >
                <tr>
                    <th class="oraGridColumnHeader" nowrap="nowrap">
                        <span oraLabel="contactInformation/type></span>
                    </th>
                    <th class="oraGridColumnHeader" nowrap="nowrap">
                        <span oraLabel="contactInformation/number"></span>
                    </th>
                </tr>
            </thead >
            <tbody>
                <tr>
                    <td class="oraNormalAlt oraDisplayCell">
                        <span oraField="type"></span>
                    </td>
                    <td class="oraNormal oraDisplayCell">
                        <span oraField="number"></span>
                    </td>
                </tr>
            </tbody>
        </table>
    </td>
</tr>
...
<xml>
    <root>
        <address> 123 Main St</address>
        <city>Alameda</city>
        <state>CA</state>
```

Copyright © 2024, Oracle and/or its affiliates.

```
        <zip>94770</zip>
        <contactInformation>
            <type>Home Phone</type>
            <number>510-555-2287</number>
        </contactInformation>
        <contactInformation>
            <type>Cell Phone</type>
            <number>510-555-4285</number>
        </contactInformation>
    </root>
</xml>
```

Embedding the Table

The embedded table is included within the overall table structure. The colspan attribute ensures that the embedded table can span the standard two columns of the overall layout table.

```
...
<tr>
    <td colspan="2" class="oraEmbeddedTable">
    -...
    -...
    -...
    </td>
</tr>
...
```

Embedded Table Structure

The embedded table is very similar to the basic layout table.

```
...
<table oraList="contactInformation" cellspacing="2">
<thead>
          -...
          -...
</thead>
<tbody>
          -...
          -...
</tbody>
</table>
...
```

- The <table> tag has a slightly smaller cellspacing and it defines the "list" element contained in the XML that will be used to provide the data.
- The <thead> element is used to give the embedded table headings for the columns.
- The <tbody> element is the element that will be repeated for each referenced "list" element in the XML. In the previous example, there are two "contactInformation" list elements, so the displayed embedded table will have two rows.

Column Headings

Embedded tables should have headings for the displayed columns. The <thead> tag defines these.

```
...
<thead>
    <tr>
        <th class="oraGridColumnHeader" nowrap="nowrap">
            <span oraLabel="contactInformation/type></span>
        </th>
```

Copyright © 2024, Oracle and/or its affiliates.

```
        <th class="oraGridColumnHeader" nowrap="nowrap">
            <span oraLabel="contactInformation/number"></span>
        </th>
    </tr>
</thead>
...
```

The "nowrap" attribute prevent the column heading from taking multiples lines. If multiples lines are required, the "nowrap" may be removed.

Input Fields

Embedded tables may be used for input as well as display only. The framework provides a convenient control to assist in the creation of editable embedded tables.

```
...
<tr>
    <td colspan="2" class="oraEmbeddedTable">
        <table oraList="contactInformation" cellspacing="2">
            <thead >
                <tr>
                    <th class="oraGridColumnHeaderButton"></th>
                    <th class="oraGridColumnHeaderButton"></th>
                    <th class="oraGridColumnHeader" nowrap="nowrap">
                        <span oraLabel="contactInformation/type></span>
                    </th>
                    <th class="oraGridColumnHeader" nowrap="nowrap">
                        <span oraLabel="contactInformation/number"></span>
                    </th>
                </tr>
            </thead >
            <tbody>
                <tr>
                    <td oraType="addGridRow"></td>
                    <td oraType="deleteGridRow"></td>
                    <td>
                        <input type="text" oraField="type"/>
                    </td>
                    <td>
                        <input type="text" oraField="number"/>
                    </td>
                </tr>
            </tbody>
        </table>
    </td>
</tr>
...
```

There are two new columns added to the input embedded table.

- oraType="addGridRow" will add a "+" button to the row. This will allow the User to add an additional row to the existing grid.
- oraType="deleteGridRow" will add a "-" button to the row. This will allow the User to delete an existing row from the grid.

Note: The <thead> tag also requires these two new columns to be added.

These controls are, as standard, placed at the beginning of the row in the order shown. Either of the controls may be omitted if required (if, for example, Users are not permitted to delete information).

The presence of either of these controls will activate the "empty list" process. This means that if the XML has no data for the "list" specified, the input grid will display with an empty row ready for the input of new information.

Copyright © 2024, Oracle and/or its affiliates.

Action Buttons

Example Action Button HTML

Action buttons are used to perform some specified function from the UI Map. The actions are as varied as the information being displayed/updated. Below are two common examples:

- Save. Normally used on an Input UI Map to allow a User to save any changes they have made.
- Cancel. Normally used on an Input UI Map to allow a User to cancel changes in progress.

```
...
<tr>
  <td colspan="2" class=" oraEmbeddedTable">
    <table cellspacing="2">
      <tr>
        <td>
          <input class="oraButton" oraMdLabel="C1_SAVE_LBL" type="button"
              onClick="oraSubmitMap('OK');"/>
        </td>
        <td>
          <input class="oraButton" oraMdLabel="CANCEL_LBL" type="button"
        onClick="oraSubmitMap('CANCEL',false);"/>
        </td>
      </tr>
    </table>
  </td>
</tr>
...
```

Button Standards

The following points highlight some standards related to buttons.

- Buttons are included as an embedded table.
- Buttons should be grouped together. They should not be placed in different areas of the UI Map.
- The location of the buttons depends mainly on the type of UI Map.
  - Display Only UI Maps should have a Record Actions section in the upper right section of the UI map.
  - Input UI Maps should have the buttons at the foot of the UI Map (after all input fields).

Available Styles

Styles are all contained in the referenced CSS stylesheets. They are applied by the HTML "class" attribute. The actual style settings used are not documented here as they may be adjusted. This section only specifies when a particular style should be used.

Note: The "class" attribute may reference more than one style (class="oraLabel oraSectionEnd")

Style	Comments	Example
oraButton	Applied to <input> elements where the type is button.	... <input class="**oraButton**" - oraMdLabel="CANCEL_LBL" - type="button" - onClick="oraSubmitMap('CA ...
oraDisplayCell	Applied to the <td> tag of an embedded table. It defines how the table cell looks (not the data contained inside the cell).	... <td class="**oraDisplayCell**

Copyright © 2024, Oracle and/or its affiliates.

Style	Comments	Example
		`  </td>` `...`

oraEmbeddedTable	Applied to the <td> tag that will contain the embedded table.	`...`
		`<tr>`
		`  <td colspan="2" class=" c`
		`  <table cellspacing="2"`
		`  -...`
		`  -...`
		`  -...`
		`  </table>`
		`  </td>`
		`</tr>`
		`...`

oraError	This style is applied to elements that are identified as "error elements". Refer to Display Errors for more information. **Note:** This style is not normally applied directly in the UI Map HTML	

oraErrorText	This style is applied to the elements concerned with error messages.	`...`
		`<table width="100%" cellpac`
		`  <tr class="oraErrorText">`
		`  <td><a href="" -`
		`  onclick="oraShowErro`
		`  return false;">`
		`  <span class="oraErro`
		`  oraErrorVar="ERRMSG-`
		`  </a>`
		`  </td>`
		`  </tr>`
		`</table>`
		`...`

oraGridColumnHeader	This style is applied to the <td> tags that define column headers within embedded table.	`...`
		`<thead >`
		`  <tr>`
		`  <th class="oraGridColu`
		`  <th class="oraGridColu`
		`  <th class="**oraGridColu`**
		`  nowrap="nowrap">`
		`  <span oraLabel="conta`
		`  </span></th>`
		`  <th class="**oraGridColu`**
		`  nowrap="nowrap">`
		`  <span oraLabel="conta`
		`  </span></th>`
		`  </tr>`
		`</thead >`

Copyright © 2024, Oracle and/or its affiliates.

Style	Comments	Example
		...
oraGridColumnHeaderButton	This style is applied to the <td> tags that define the column headers for the "+" and "-" buttons used on editable embedded tables.	... <thead > <tr> <th class="**oraGridColu** </th> <th class="**oraGridColu** </th> <th class="oraGridColu <span oraLabel="conta </th> <th class="oraGridColu nowrap="nowrap"> <span oraLabel="conta </th> </tr> </thead > ...
oraInput	This style is applied to input fields: • <input type="text">• <input type="checkbox">• <select>• <textarea> **Note:** This can normally be omitted as input styles are applied automatically when oraSchemaDataTypes="true".	... <input type="text" class=" oraField="address"/> ...
oraInputMoney	This style is applied to input fields: • <input type="text">• <select> (rare)• <textarea> (not recommended) **Note:** This can normally be omitted as input styles are applied automatically when oraSchemaDataTypes="true".	... <input type="text" class=" oraField="amount"/> ...
oraInputNumber	This style is applied to input fields: • <input type="text">• <select> (rare)• <textarea> (not recommended) **Note:** This can normally be omitted as input styles are applied automatically when oraSchemaDataTypes="true".	... <input type="text" class=" oraField="count"/> ...
oraLabel	This style is applied to standard label fields that are right aligned.	... <td class="**oraLabel**" - oraLabel="address"></td>

Copyright © 2024, Oracle and/or its affiliates.

Style	Comments	Example
	Note: This can normally be omitted as it is applied by the <col> tag.	...
oraLabelAlt	This style is applied to standard label fields only if it is desired to have the label aligned to the left.	... <td class="**oraLabelAlt**" - oraLabel="address"></td> ...
oraLabelCenter	This style is applied to standard label fields only if it is desired to have the label aligned in the center of the cell.	... <td class="**oraLabelCenter**" oraLabel="address"></td> ...
oraLink	This style is applied to foreign key references (links). This is automatically added by the UI Map framework but can also be used manually if desired.	... <td class="oraLabel"> <a href="www.google.com" class="oraLink">Google</a ...
oraNormal	This style is applied to standard data fields (display only) that are left aligned. **Note:** This can normally be omitted as it is applied by the <col> tag.	... <td class="**oraNormal**" - oraField="address"></td> ...
oraNormalAlt	This style is applied to standard data fields (display only) only if it is desired to have the data aligned to the right.	... <td class=" **oraNormalAlt**" - oraField="address"></td> ...
oraNormalCenter	This style is applied to standard data fields (display only) only if it is desired to have the data aligned in the center of the cell.	... <td class=" **oraNormalCenter** oraField="address"></td> ...
oraPageTitle	This style is applied to the element that contains the page title. **Note:** This style is not normally applied directly in the UI Map HTML. The is created by the UI Map framework when the UI Map is displayed in the page area.	... <span class=" **oraPageTitle** oraMdField="PAGE_TITLE_LE ...
oraSectionEnd	This style is applied to the <td> tags at the end of a "section" (group of elements). It provides some space to separate the section from the following information.	... <tr> <td class="**oraSectionEnd** oraLabel="zip"> </td> <td class="**oraSectionEnd** oraField="zip">>

Copyright © 2024, Oracle and/or its affiliates.

Style	Comments	Example

	Note: The style must be applied to both <td> tags or the label may be misaligned with the data/input.	`  </td>`
		`</tr>`
		`...`

oraSectionHeader	This style is applied to the <td> tag used to give a heading for a section within the information being displayed. It does not provide spacing before or after itself. The oraSectionStart and oraSectionEnd classes are used for this.
	Note: The section header should span both the label column and the data column.
		`...`
		`<tr>`
		`  <td class="oraSectionHea`
		`  colspan="2" -`
		`  oraMdField="DATA_SECTI`
		`</tr>`
		`...`

oraSectionStart	This style is applied to the <td> tags at the start of a "section" (group of elements). It provides some space to separate the section from the previous information (often a section header).
	Note: The style must be applied to both <td> tags or the label may be misaligned with the data/input.
		`...`
		`<tr>`
		`  <td class="oraSectionSta`
		`  oraLabel="zip"></td>`
		`  <td class="oraSectionSta`
		`  oraField="zip"></td>`
		`</tr>`
		`...`

oraTableData	This style is applied to the <col> tag for the data column of the main table (second column). It is used to provide a percentage width for the horizontal space to be used for the information.	`...`
		`<colgroup>`
		`  <col class="oraLabel`
		`  oraTableLabel"/>`
		`  <col class="oraNormal`
		`  oraTableData"/>`
		`</colgroup>`
		`...`

oraTableLabel	This style is applied to the <col> tag for the label column of the main table (first column). It is used to provide a percentage width for the horizontal space to be used for the labels.	`...`
		`<colgroup>`
		`  <col class="oraLabel -`
		`  oraTableLabel" -/>`
		`  <col class="oraNormal ora`
		`</colgroup>`
		`...`

oraTinyText	This style is typically applied directly beneath an <input> tag to provide information or a hint to the user concerning information relevant to the input. For example, name or address format.	`...`
		`<tr>`
		`  <td oraLabel="address"><`
		`  <td>`
		`  <input type="text" ora`
		`  <div class="oraTiny`
		`  oraField="addressFor`
		`  </td>`
		`</tr>`
		`...`

oraZoneMap	This style is used applied when the UI Map is to be displayed as a zone on a portal.	`...` `<body class="oraZoneMap">`

Copyright © 2024, Oracle and/or its affiliates.

Style	Comments	Example
		...

Using OJET

There are some UI maps delivered by the product that use UI widgets provided by Oracle JavaScript Extension Toolkit (OJET). Releases for OJET do not always align with releases of the framework. In addition, there are times when OJET adjusts APIs that the product uses. The framework will attempt to ensure that each release of the product has the latest and greatest version of the OJET libraries. Implementations are discouraged from attempting to use features in OJET that are not used by the product because the product is not necessarily testing those features and is not ensuring that upgrades to the APIs for those features are backward compatible.

Note that the product isolates the references to OJET into a UI map fragment that is included in the maps that use OJET widgets. This is so that changes to future versions of OJET are minimized to a single place. The map is called F1–OJETLIBS. If your implementation wants to use OJET, the recommendation is to use this UI map fragment.

Ensuring Unique Element IDs for UI Maps

The following describes how to modify JavaScript code to ensure the proper rendering of unique element IDs for UI Maps.

The modification is required only for code that renders HTML using a getElementById() (or similar) function to generate list IDs and avoid account verification or related errors.

The following sample snippet contains the necessary modifications:

```
...
function getElementsFromList(namePrefix) {
    var ret = [];
    var elements = document.getElementsByTagName("INPUT");
    for(var i=0;i<elements.length;i++) {
        var elemID = elements[i].id;
        if((id) && (id.startsWith(namePrefix + -'_')) {
            ret.push(elements[i]);
        -}
    -}
...
return ret;
```

Since IDs aren't necessarily unique in generated UI Map IDs, the code shown above ensures uniqueness at runtime by appending an underscore and row number (e.g., myField_1, myField_2) for proper handling by Framework in the rendered HTML, while still allowing you to reference the unmodified IDs contained in the generated UI Map.

A switch in the spl.properties file also permits you to disable the generation of unique IDs for elements in a grid (as described below), though, for standards compliance reasons, it is highly recommended that this switch be left at is default value.

```
 -
Property Name: spl.runtime.compatibility.uiMapDisableGenerateUniqueHtmlIDs
File Name: spl.properties (under web project in FW)
Default Value: false
Accepted Values: true or false
Description: This property controls the generation of unique IDs for all -
input elements inside -
a list. When this value is set to true it disables the generation of -
unique IDs, thus -
```

Copyright © 2024, Oracle and/or its affiliates.

```
replicating the old behavior. When this property is set to false or this -
property is missing -
it enables the generation of unique IDs, thus enabling the list to be -
standards-compliant.
```

Process Flows

This section describes concepts and provides guidelines related to the configuration of various type of process flows.

Understanding Process Flows

A process flow is a user interface guiding a user through a series of actions in order to accomplish a specific task. The task can be as simple as the collection of information in order to update business data or involve more complex logic such as submitting and tracking batch processes, exchanging messages with an external system, etc.

This section describes topics related to designing and working with process flows.

A Process Flow Is Made Of Panels

Each process flow consist of a number of sequential "steps" needed to accomplish a certain task. Each step is represented by a stop on a progress bar and an associated panel.

Progress is linear in that each step may only lead to a single next step. A process flow always starts at a single initial step but allows for one or more final steps at which the user may choose to complete the process.

At any step the user may take the following actions:

- Enter data or take action as prompted by the step’s panel.
- Continue to a next step, if any.
- Navigate back to any step they have previously visited.
- Finish the process flow if the current step is a final step.
- Save off their work on the process flow and either continue their work or navigate away. At a later time the user may resume their work on the process flow from where they have left off.
- Cancel the process flow.

A process flow type defines the entire metadata needed to control the behavior of process flows of a given type. This includes the sequence of steps, the panel and rules associated with each step and more.

Panel Presentation

The panel presented on each step of the process flow is rendered based on the panel type selected on the step configuration.

For a **Data Area** type of panel, the panel is rendered using UI Hints built into the schema of a specified data area.

For a **UI Map** type of panel, the panel is rendered using a specified UI map fragment.

For a **Panel Set** type of panel, the panel is rendered as a set of tabs, one for each panel that references the current panel as its parent panel. Clicking on each tab renders the UI based on the selected panel’s type. The behavior of a panel set is analogous to or can be thought of as a nested process flow within a single panel. A panel set may not include another panel set.

A Single UI Map

While the user enters data one panel at a time, data is collected and captured at the entire process flow level. All the elements edited and displayed on the various panels of the process flow must be included in the overall process schema and referenced by their appropriate XPath location in that schema. Each type of process flow defines its unique schema on a designated UI map.

Copyright © 2024, Oracle and/or its affiliates.

The UI map's HTML should simply include the **Process Flow Controller (F1-ProcessController)** UI map fragment and nothing else. The latter is designed to render the user interface for a process flow based on the metadata defined on its process flow type.

Panel Scripts

A single **Pre-Processing** service script may be associated with a panel for the purpose of preparing the data before the panel is presented to the user. The system calls the script each time the user navigates to this panel on a non-finalized process flow.

A single **Post-Processing** service script may be associated with a panel for the purpose of validating the data entered by the user on that panel. The system calls the script each time the user exits the panel on a non-finalized process flow.

Process Flow Scripts

A single **Process Start** service script may be associated with a process flow for the purpose of preparing the data before a newly initiated process flow is presented to the user.

A single **Process End** service script is executed when the user clicks on the **Finish** or **Cancel Process** buttons to complete the process flow. The entire set of data collected by the process flow UI map schema is provided to this service script for final processing.

Note: The schema of these scripts should be the same as the **Process Flow UI map** schema. This would ensure that business rules have access to the entire data captured by the user.

Summary Panel

By default, when a user finishes a process flow, as well as when they view an already closed process, the process flow portal displays the last panel the user was working on. For more complex process flows, you may provide a panel that summarizes information from the entire process. When specified on the process flow type, the system displays that summary panel instead a the default view of a closed process. The user may toggle between the summary view and the detailed view at any time.

The summary panel may be implemented as a UI map fragment or a data area like other panels but may not be associated with any panel script.

Launching A New Process Flow

Each process flow must be initiated and executed by a designated BPA script that has the following simple steps:

- Set BPA area height to zero.
- Populate the temporary variable **$processCode** with the code of the specific process flow type.
- Transfer control to the **Process Flow Navigation (F1-PROCEXEC)** common BPA script.

Launching a process flow of a specific type is assumed to be made from a designated menu option or within a specific context applicable to that type of work. As such, this item needs to be configured specifically for each type of process flow. The navigation option associated with that launching option should reference the BPA script associated with the corresponding type of process flow.

Saving Off Work

The user may save off their work on a process flow at any time. If enabled on the process flow type, the system would also automatically save the data when the user navigates to another panel or away from the process flow portal.

On either manual or automatic save operations, the **Process Manager** script defined on the process flow type is called to save off the entire process flow data on a designated record in the database.

The base product provides a generic **Process Flow** maintenance object that is designed to support the storage of any type of process flow as well a corresponding process manager script. Should process flows of a specific type be stored elsewhere, then a dedicated process manager script should be implemented and used instead.

Copyright © 2024, Oracle and/or its affiliates.

Review Process Flow Records

A process flow record is created when a user saved off their work or the process flow type enables automatic saving. When the user finishes a process flow, depending on the process flow type configuration the record may be deleted or retained for audit purposes.

Reviewing process flow records of a specific type is assumed to be made from a designated menu option or within a specific context applicable to that type of work. As such this item needs to be configured specifically for each type of process flow. The navigation option associated with that item should reference a query portal designed to query and manage the specific type of process flow records.

In addition to standard query features, the query portal should allow the user to resume their work on a non-finalized process flow as well as review the data captured on a completed record. When implementing such a query portal, refer to the sample **Process Flow Query (F1PRSTRQ)** portal for key features. Your specific product may already include such query portals for process flow types it supports. Refer to your specific product documentation for additional information.

Designing Process Flows

A wide range of process flows may be designed to implement online tasks business users may need to perform. Refer to Understanding Process Flows for more information.

Below is a high-level summary of the steps required to design and configure various types of process flows.

Analyze The Task

The following provides a the high-level guidance in analyzing the necessary data and rules needed to support an implementation of a new type of process flow:

- Identify the data and business rules that are needed to complete the entire task.
  - When possible, organize data into logical groups of details defined as data areas.
  - Design logic for the final processing script that accepts the entire data entered along with the user's action to either complete or cancel the task.
  - If data needs to be initially prepared for a newly initiated process flow, design logic for a corresponding initial processing script.
  - Determine whether a summary panel may be useful for the user to review the outcome of a completed process.
- Design the sequence of panels the user should follow to complete the task. Identify the initial and final panels.
- Review each panel and determine the following:
  - Is it a single panel or a nested process flow (i.e. a panel set)?
  - What is the title of the panel and its label on the progress bar?
  - How should the layout be rendered? Use a data area that leverages UI Hints to describe the panel layout when possible. For more complex HTML requirements you may use a UI map fragment.
  - Should details on this panel be defaulted with some initial values when the user enters this panel? If so, design logic for a corresponding pre-processing script.
  - If data entered on this panel should be validated before the user exits the panel, design logic for a corresponding post-processing script.
- Design the summary panel if needed.
- Determine whether this type of process flow should be automatically saved by the system or solely rely on the user to control when data is saved.
- Determine whether a record of a completed process flow should be retained for audit purposes or deleted upon completion.
- Determine whether a user may delete in progress records of such process flow.
- Determine whether process flows of this type are saved and stored in a dedicated table in the database or are saved in the generic table provided by the base product. The assumption is that introducing new designated storage maintenance objects is not common. Refer to your edge product documentation for additional maintenance objects that may have been provided to support specific types of process flows shipped with the product.

Copyright © 2024, Oracle and/or its affiliates.

- Determine the context in which a user can initiate, resume work on, and review process flows of that type.
  - The process flow may be launched from a menu or a favorite script etc.
  - Design the query portal a user would use to find and resume a saved off process flow of this type as well as review completed process flow records. Depending on the context of this task, the query may be associated with the same menu line that initiates the process flow.

Using UI Map Panels

Typically a map fragment is rendered as part of a complete UI map but when used as a process flow panel the system renders the panel assuming the fragment contains a complete map.

Therefore, such fragment should includes the following items:

- The standard error message section used for reporting errors.
- This line <xml style="display:none"></xml> at the end of the HTML content for proper binding of data to their HTML elements.

A UI map based panel allows for more control over some of the buttons available to the user. For example, the panel may prevent the user from proceeding to the next panel unless required information is entered. Refer to the explicit APIs listed in the process controller UI Map (F1-ProcessController) for more information.

Configure The Process Flow Type

The following describes the high-level steps needed to configure a new type of process flow:

- Create the UI map for the process flow. Its schema should contain all the details needed to complete the task. Its HTML should only include the process controller UI map fragment.
- Create a start processing service script if needed.
- Create a final processing service script such that it applies all the rules needed to complete or cancel the task.
- For each panel perform the following:
  - Create a UI map fragment for it unless it corresponds to a data area already included in the process flow schema. Make sure the XPath of each element references in the data area or UI map is the same XPath of that element in the process flow's UI map schema.
  - Create a pre-processing service script if needed. Not applicable to the summary panel.
  - Create a post-processing service script if needed. Not applicable to the summary panel.
  - Set up a label and title MD fields if existing ones cannot be used.
- Create a BPA script for the process flow.
- Configure the process flow type record.
- Set up a navigation option and reference the BPA script on it. Associate the navigation option with a menu or provide other means for the user to initiate the process flow.
- Create a query portal for the specific type of process flow records. Add the new query portal to a menu or provide other means for the user to resume work on a process flow they have saved off or review completed records.

Control Visibility of Standard Buttons

You may leverage the following javaScript functions to control the visibility of standard process flow buttons:

- Next: **enableNext()**, **disableNext()**, **hideNext()** , **showNext()**

  .
- Finish: **enableFinish()**, **disableFinish()**, **hideFinish()** , **showFinish()**

  .
- Save: **enableSave()**, **disableSave()**, **hideSave()** , **showSave()**.

Copyright © 2024, Oracle and/or its affiliates.

Defining Process Flow Types

This portal is used to maintain process flow types.

Refer to Understanding Process Flows for more information.

You can access the portal from the**Admin Menu** > **System** > **Process Flow Type.**

The following zones may appear as part of the portal's **Main** tab page

- **Process Flow Type List.** This zone lists all process flow types. Broadcast a record to display the details of the selected process flow type.
- **Process Flow Type.** This zone provides information about the selected process flow type.

Action Method

Action method is an object that may be used to define configuration based on some criteria. For example, when initiating a Start Service scenario, the service request type may differ based on the CIS division and the customer class. When a user initiates a Start Service process, the system can prompt the user for information about the customer that can determine the CIS division and the customer class. The system can then call an algorithm passing this information to retrieve the appropriate service request type to use as defined in the action method. Throughout this section, the term 'criteria' is used to identify the information used to determine the desired configuration and the term 'details' or 'results' is used to identify the configuration information being determined.

If you were to design a use case to use this functionality consider the following:

- What information needs to be identified (the 'details'/ 'results')? The example above uses a configuration object as its example (service request type). However, the information you are determining could be anything that can differ by some combination of attributes. You could use this to determine which collection algorithm to execute based on criteria. Or the number of days to wait to continue with some process for some use case.
- When and where do I need to determine this information and what do I do with it once I receive it?
  - Is this needed as part of a user interaction?
  - Or is this needed for internal processing?
- What is the criteria needed to determine the desired information? Do I have this information at the time that I need it? Or do I need to prompt a user for the information?

The framework product supplies the generic objects needed to support this type of functionality. The individual edge products or implementations can build their use cases with the tools provided. The topics in this section further describes the provided features.

Action Method Role

The action method functionality is driven by the Action Method Role, which represents a given use case. Some examples of action method roles that may be relevant in an enterprise product are 'Start Service', 'Stop Service' and 'Transfer Service'. The valid action method roles are defined using an extendable lookup and referenced on the Action Method. One and only one action method may be defined for one action method role.

The action method role record may reference an action method role BPA script, if the use case for this Action Method Role involves user interaction. The responsibility of this script, if needed is to take the action method role value, determine the action method that references the role and identify the more specific Action Processing BPA script (defined on the business object) and transferring control to that script. This reference may be used for audit or information purposes, depending on how you have implemented the user experience. The action method role BPA is ideally configured on an appropriate menu entry or an action button on a specific user interface related to this use case.

If your business use case does not require user interaction, then an action method role BPA script is not necessary.

Copyright © 2024, Oracle and/or its affiliates.

Action Method Business Object

The action method's business object is used to define the information that needs to be configured for both the criteria and the results. Continuing to use the above example, the business object for a Start Service use case defines a list that captures the CIS division and customer class along with the service request type to use. The information is configured in the XML data area for the action method and the schema design is dictated by the business requirements. For example, it may be that your use case dictates a default value for the results (in our case, the service request type) plus override values based on a division and customer class combination.

Get Action Method Details Plug-in

Besides defining the schema information for capturing the criteria and results, the business object defines the algorithm that should be invoked to retrieve the results based on the criteria.

The plug-in spot is **Get Action Method Details**. Its API is flexible in the information it receives and returns so that each action method use case can design the algorithm based on its business requirements.

The algorithm receives a list of 'Criteria Data' using a 'name' and up to 5 values (to handle multi-part primary keys). Typically only 'value 1' is needed for each entry. The algorithm type designed for a given action method business object can determine what information is receives. It could be that it expects the criteria elements defined on the action method. In our example, the algorithm type could expect to receive the CIS division and customer class. Alternatively, the algorithm type could be designed to receive a value that can be used to determine the criteria needed. For example, it could receive the account ID and use that information to determine the CIS division and customer class.

The algorithm returns one or more results as 'Details'. The information returned is based on the use case. In our example, the service request type is returned. Your business requirements may use the action method role to return multiple results, for example a service request type and a process flow type to invoke to capture all the information needed to start service.

The product provides a business service **F1-RetrieveActionMethodDetails** to call the algorithms for a given action method value.

Action Processing BPA

If your business requirement requires user interaction either to determine the criteria to pass to the Get Action Method Details algorithm or to use the resulting information to continue with an online action, an Action Processing BPA is needed. This BPA should be designed and linked to your action method business object as an option using the **Action Processing BPA Script** option type.

The design of this BPA depends on your business requirements. But at a high level, it may need to prompt the user for information needed to determine the criteria associated with the action method details, then call the **F1-RetrieveActionMethodDetails** business service to execute the algorithms for your action method. Once the results are returned, the script would proceed with the next step in the business use case.

Note: If any additional information needs to be gathered by the server before or after the call to the business service that executes the algorithms, then the recommendation is that the BPA should call a service script that does all the server level logic at one time.

Action Method Role vs. Action Method Business Object

The action method role defines the use case. The business object is used to design the schema for the criteria and the results and identifies the action processing BPA and the Get Action Method Details plug-in. It may warrant a unique business object for each action method role. However, there may be cases where more than one action method role and its action method can reuse the same business object. For example, imagine there are different service request types for Start Service, Stop Service and Transfer Service, however in all cases the valid value is driven by the CIS division and customer class. These three separate action method roles and action methods can reuse a common business object defining the criteria of CIS division and customer class determining the appropriate service request type.

Copyright © 2024, Oracle and/or its affiliates.

Defining Action Methods

Refer to Action Method for an overview of action method functionality.

To maintain the action methods applicable to your implementation, open **Admin** > **System** > **Action Method.**

This is a standard All-in-One portal and includes the standard List and display zones for an action method.

The information captured on the action method depends on the business objects supported by your product or implementation. Refer to the embedded help text for more information.

Algorithms

In this section, we describe how to set up the user-defined algorithms that perform many important functions including:

- Validating the format of a phone number entered by a user.
- Validating the format of a latitude/longitude geographic code entered by a user.
- In products that support payment and billing:
  - Calculating late payment charges.
  - Calculating the recommended deposit amount.
  - Constructing your GL account during the interface of financial transactions to your GL
- And many other functions...

The Big Picture Of Algorithms

Many functions in the system are performed using an algorithm. Algorithms are called at strategic points (often referred to as a "plug-in spot") within system processing to perform specific code. The plug-in spots are preconfigured with the product and vary from object to object. Algorithms are used by the base product to provide out of the box functionality. The algorithm framework provides a means for implementations to insert their own custom code, which can be instead of the base delivered code or in addition to it.

Note: Algorithm = Plug-in. We use the terms plug-in and algorithm interchangeably throughout this documentation.

Algorithm Type Versus Algorithm

Often the system will just talk about algorithms but in actuality there are two important objects that provide this plug-in functionality.

- An **Algorithm Type** defines the plug-in spot. In addition, it defines program to execute. The program may be defined using Java or using a plug-in script. The algorithm type also defines any parameters that must be supplied to algorithms of this type.
- An **Algorithm** is an instance of an **Algorithm Type**. If the algorithm type does not define any parameters, then there will be only one algorithm for the algorithm type. If the algorithm defines parameters, then there could be several algorithms for a given algorithm type, one for each unique set of parameter values for a given use case. The algorithm is the object that gets referenced on a given control table when configuring your system.

Algorithm Plug-in Spot

The plug-in spot, also called Algorithm Entity and System Event, defines several aspects of the functionality for algorithms that are executed.

- When in system processing are algorithms of this plug-in spot executed? For example, algorithms for the Installation Option - To Do Pre-creation are called when a To Do is getting added, before any insert to the database is performed.
- The API for the plug-in spot. This is often referred to as the "hard parameters" for the plug-in spot. What information is every algorithm provided when called? Additionally, if the algorithm is expected to return information to the

Copyright © 2024, Oracle and/or its affiliates.

caller, this is also part of the hard parameters. For example, the Installation Option - To Do Pre-creation plug-in spot receives all the information about the To Do that is about to be added. It may add or update values. It may also return an indication to not create the To Do, if desired.

> **Note:** The hard parameters are different from the parameters defined on the algorithm type. Those are referred to as the "soft parameters". Those parameters are used to configure settings that the algorithm type code uses per its specific functionality.

- What is the responsibility of algorithms for the plug-in spot? When defining a plug-in spot, the product defines the responsibility of the code for the algorithm types created for that spot. For example, is the algorithm determining information and returning it? Or is it supposed to perform an update? Algorithms for the Installation Option - To Do Pre-creation plug-in spot should not do any updates. Rather they should adjust the To Do information (if desired) in the hard parameters and return them. The calling logic will use that adjusted information when inserting the To Do record.
- Where do you configure the algorithm? We have been using the Installation Option - To Do Pre-creation plug-in spot as an example. As the description suggests, these algorithms are configured on the Installation Option - Algorithm collection. Each plug-in spot defines where (on which control table) the algorithms are plugged in. The system standard for the description of the Algorithm Entity flag value that defines the available plug-in spots is "Control Table Name - Event". For example the To Do Type - Calculate Priority algorithm plug-in spot means that algorithms are configured on the To Do Type. This information also indicates the granularity of the algorithm. Installation Option algorithms are system wide. But an algorithm plugged in on To Do Type, for example, indicates that only To Do Entries for that To Do Type will have that algorithm executed.
- Are multiple algorithms allowed? There are some plug-in spots where it doesn't make sense to support multiple algorithms. An example of this is the Batch Control - Select Records plug-in. This plug-in is used by the plug-in driven batch process and is responsible for selecting the records to process. Only one algorithm of this type is allowed on a given Batch Control. An example of a plug-in spot that would support multiple algorithms is a validation plug-in spot. Note that when a plug-in spot supports multiple algorithms, there can be yet additional aspects of the plug-in spot design.
  - In some cases, this type of plug-in spot simply executes each algorithm. A validation plug-in spot is an example of this.
  - In some cases, the algorithm could return an indication to the plug-in spot 'driver' to not execute any subsequent algorithms. Business Object Status - Enter is an example of a plug-in spot that supports this capability.
  - In some cases, the plug-in spot may support sharing information from one algorithm to the next. Migration Plan - Pre-Compare is an example of a plug-in spot that supports this capability.

Each plug-in spot is defined in the Algorithm Entity lookup (**ALG_ENTITY_FLG**). When defining an algorithm type or defining a plug-in script, once you select the algorithm entity value, a help icon should appear. Clicking this icon will show you the description of the plug-in spot. It may be that online help provides more information about a given plug-in spot as well. Note that to see the hard parameters defined for a given plug-in spot, go to the Script page in add mode and select the script type of plug-in and the desired algorithm entity lookup value. Then navigate to the Data Area tab. The **View Plug-In Script Data Area** link will display the hard parameters.

Base Product Algorithms

The base product provides many, many algorithm types and algorithms out of the box. Sometimes algorithms are delivered to provide support for common use cases that we know our clients will need. Many algorithms, however, are provided because we are implementing system functionality using the same infrastructure and capabilities that our implementation use for their own custom code.

When an algorithm type is delivered, the decision as to whether an algorithm is also provided depends on some factors related to whether soft parameters are defined for the algorithm type.

- If no parameters are defined for the algorithm type, a corresponding algorithm is typically also provided.
- If parameters are defined and the values are on configuration data that is normally owned by the implementation, a base algorithm will probably not be defined.
- If parameters are defined and the values are not dependent on configuration data, it's possible that one or more algorithms are provided.

Copyright © 2024, Oracle and/or its affiliates.

To review the algorithm types that are delivered with the product, go to the Algorithm Type query. The query provides several search options, including searching by the algorithm entity lookup value.

How To Add A New Algorithm

Before you can add a new algorithm, you must determine if you can use one of the algorithm types supplied with the system by using the algorithm type query.

If you can use one of the algorithm types, simply add the algorithm and then reference it on the respective control table. Refer to Defining Algorithms for how to do this.

If you have to add a new algorithm type, you may have to involve a programmer. Let's use an example to help clarify what you can do versus what a programmer must do. Assume that you require an additional geographic type validation algorithm. To create your own algorithm type you must:

- Write a new program to validate geographic type in the appropriate manner. Alternatively, you may configure a plug-in script to implement the validation rules. The advantage of the latter is that it does not require programming. Refer to plug-in script for more information.
- Create an Algorithm Type called **Our Geographic Format** (or something appropriate). On this algorithm type, you'd define the name of the program (or the plug-in script) that performs the function. You'd also define the various parameters required by this type of algorithm.
- After creating the new Algorithm Type, you can reference it on an Algorithm.
- And finally, you'd reference the new Algorithm on the Geographic Type that requires this validation.

Minimizing The Impact Of Future Upgrades

The system has been designed to use algorithms so an implementation can introduce their own logic in a way that's 100% upgradeable (without the need to retrofit logic). The following points describe strong recommendations about how to construct new algorithm type programs so that you won't have to make program changes during future upgrades:

- Do not alter an algorithm type's hard parameters. For example, you might be tempted to redefine or initialize parameters defined in an algorithm type's linkage section. Do not do this.
- Follow the naming conventions for the new algorithm type code and your source code, i.e., both the source code and the algorithm type should be prefixed with "CM". The reason for this naming convention is to make it impossible for a new, base-package algorithm type from overwriting your source code or algorithm type meta-data (we will never develop a program or introduce meta-data beginning with CM).
- Avoid using embedded SQL to perform insert/update/delete. Rather, invoke the base-package's object routines or common routines.
- Avoid using base messages (outside of common messages, i.e., those with a message number < 1000) as we may deprecate or change these messages in future releases. The most common problem is caused when an implementation clones a base package algorithm type program because they need to change a few lines of logic. Technically, to be 100% upgradeable, you should add new messages in the "90000" or greater category (i.e., the category reserved for implementation-specific messages) for every message in your new program even though these messages may be duplicates of those in the base package.

Defining Algorithm Types

The system provides many algorithm types to support base product functionality. If you need to introduce a new type of algorithm, open **Admin** > **System** > **Algorithm Type**.

Use the **Algorithm Type Query** portal to search for an existing algorithm type. Once an algorithm type is selected, you are brought to the maintenance portal to view and maintain the selected record.

Fastpath: Refer to The Big Picture Of Algorithms for more information.

CAUTION: Important! If you introduce a new algorithm type, carefully consider its naming convention. Refer to System Data Naming Convention for more information.

Copyright © 2024, Oracle and/or its affiliates.

Description of Page

Enter an easily recognizable **Algorithm Type** and **Description**.

Owner indicates if this algorithm type is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add an algorithm type. This information is display-only.

Enter a **Detailed Description** that describes, in detail, what algorithms of this type do.

Use **Algorithm Entity** to define where algorithms of this type can be "plugged in". If a detailed description about an algorithm entity is available, a small help icon is visible adjacent to the dropdown. Click the icon to view the information.

Note: The values for this field are customizable using the lookup table. This field name is **ALG_ENTITY_FLG**.

Use **Program Type** to define if the algorithm's program is written using **Java**, a **Plug-In Script**, or **Java (Converted)**, meaning the program has been converted to Java.

Note: Java (Converted) program types are not applicable to all products.

Use **Program Name** to define the program to be invoked when algorithms of this type are executed:

- If the Program Type is **Java (Converted)**, enter the name of the converted program.
- If the Program Type is **Java**, enter the Java class name.
- If the Program Type is **Plug-In Script**, enter the plug-in script name. Only plug-in scripts defined for the algorithm entity may be used.

Note: View the source. If the program is shipped with the base package, you can use the adjacent button to display the source code of this program in the Javadocs viewer. For plug-in scripts, drill into the plug-in script to view the details.

Use the **Parameter Types** grid to define the types of parameters that algorithms of this type use. The following fields should be defined for each parameter:

- Use **Sequence** to define the relative position of the **Parameter**.
- Use **Parameter** to describe the verbiage that appears adjacent to the parameter on the Algorithm page.
- Indicate whether the parameter is **Required**. This indicator is used when parameters are defined on algorithms that reference this algorithm type.
- **Owner** indicates if the parameter for this algorithm type is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add an algorithm type with parameters. This information is display-only.

Note: When adding a new algorithm type that is for a Java program, the parameters are automatically generated based on the Java code. Once an algorithm type exists, any additional parameters defined in the Java code should be manually added to the algorithm type. For other program types, algorithm type parameters must be manually defined.

Note: When a new algorithm type parameter is added for any program type, existing algorithms for the algorithm type do not automatically get updated with the new parameter. The algorithms must be manually updated.

Defining Algorithms

This portal is used to view and maintain information about an algorithm.

You can access the portal from the**Admin Menu** > **System** > **Algorithm**. You are brought to a query portal with options for searching for a specific algorithm. Once an algorithm has been selected you are brought to the maintenance portal to view and maintain the selected record.

Refer to The Big Picture Of Algorithms for more information.

Copyright © 2024, Oracle and/or its affiliates.

> **CAUTION:** Important! If you introduce a new algorithm, carefully consider its naming convention. Refer to System Data Naming Convention for more information.

The following zones may appear as part of the portal's **Main** tab page

- **Algorithm.** This zone provides information about the selected algorithm.
- **Other Algorithms.** This zone lists other algorithms of the same algorithm type. The zone is only visible when such algorithms exist.

The following zones may appear as part of the portal's **References** tab page

- **Algorithm References.** This zone list various configuration entities referencing the current algorithm.

> **Note:** If the product delivers an algorithm with parameter values defined, an implementation may override the base provided parameter values by adding an additional effective dated collection of parameters.

> **Note:** If an algorithm is defined and subsequently a new parameter is added to the algorithm type, existing algorithms for the algorithm type should be updated to add a new effective dated entry to the parameter collection. At this point the latest list of parameters for the algorithm type are visible. Configure the parameters accordingly.

Advanced Algorithm Tips

Initiating Actions that Cannot be Rolled Back

If you have an algorithm that creates a sync request or performs some other logic that could cause a message to be sent and cannot be rolled back, you should be sure to cater for warnings in the design.

- The algorithm should be performed after all validations that could accumulate warnings.
- The algorithm should include a step to call the Business Service **F1-CheckWarning** and see if any warnings have been accumulated. If so, don't perform the action. It means that the user will get a warning and if they click "OK", all the processing will happen again, but this time without accumulating any warnings. Refer to the plug-in script **F1-GnChgDCap** for an example.

Script Options

We use the term "script" to define processing rules that your implementation sets up to control both front-end and back-end processing:

- Rules that control front-end processing are defined using Business Process Assistant (BPA) scripts. For example, your implementation could set up a BPA script to guide a user through your organization's payment cancellation process.
- Rules that control back-end processing are defined using Server-based scripts. For example, your implementation could set up a server-based script to control the processing that executes whenever a given type of adjustment is canceled.

The topics in this section describe how to configure your scripts.

The Big Picture Of Scripts

This section describes features and functions that are shared by both BPA scripts and server-based scripts.

Scripts Are Business Process-Oriented

To create a script, you must analyze the steps used to implement a given business process. For example, you could create a "stop auto pay" BPA script that:

Copyright © 2024, Oracle and/or its affiliates.

- Asks the user to select the customer / taxpayer using an appropriate search page
- Asks the user to define the date on which the person would like to stop making automatic payments
- Invokes a server-based script that populates the end-date on the account's latest automatic payment instructions.

After you understand the business process, you can set up a script to mimic these steps. If the business process is straightforward (e.g., users always perform the same steps), the script configuration will be straightforward. If the business process has several logic branches, the composition of the script may be more challenging.

A Script Is Composed Of Steps

A script contains one or more steps. For example, a "stop auto pay" BPA script might have three steps:

- Ask the user to select the customer / taxpayer using an appropriate search page
- Ask the customer the date on which they'd like to stop making automatic payments (and default the current date)
- Invoke a server-based script that, in turn, updates the account's auto pay options.

Each step references a step type. The step type controls what happens when a step executes. It might be helpful to think of a script as a macro and each step as a "line of code" in the macro. Each step's step type controls the function that is executed when the step is performed.

Fastpath: Refer to How To Set Up Each Step Type for a detailed description of all available step types and how to set them up.

A Script May Declare Data Areas

Both BPA and server-based scripts may have one or more data areas:

- If the script contains steps that exchange XML documents, you must declare a data area for each type of XML document. For example, if a BPA script has a step that invokes a service script, the BPA script must declare a data area that holds the XML document that is used to pass information to and from the service script.
- You can use a data area as a more definitive way to declare your temporary storage. For example, you can describe your script's temporary storage variables using a stand-alone data area schema and associate it with your script.

Various step types involve referencing the script's data areas as well as support the ability to compare and move data to and from field elements residing in the data areas.

An Edit Data step supports the syntax to dynamically declare data areas as part of the step itself. This technique eliminates the need to statically declare a data area. Refer to Edit Data Syntax for more information on edit data commands and examples of the use of dynamic data areas.

Note: Some server based scripts may not use data areas as means of defining or exchanging data, depending on script type and the chosen scripting technique. Refer to The Big Picture Of Server Based Scripts for an overview of server scripts and their applicable scripting options.

Securing Script Execution

The system supports the ability to secure the execution of scripts by associating the script with an Application Service. Refer to The Big Picture of Application Security for more information. Application security is optional for user-invocable BPA scripts. If a script is not associated with an application service, all users may execute the script. Otherwise, only users that have **Execute** access to the application service may execute the script. For service scripts, the application service is required.

The Big Picture Of BPA Scripts

Fastpath: Refer to The Big Picture Of Scripts to better understand the basic concept of scripts.

Copyright © 2024, Oracle and/or its affiliates.

Users may require instructions in order to perform certain tasks. The business process assistant allows you to set up scripts that step a user through your business processes. For example, you might want to create scripts to help users do the following:

- Add a new person to an existing account
- Set up a customer to pay automatically
- Modify a customer who no longer wants to receive marketing information
- Modify a customer's web password
- Record a trouble order
- Merge two accounts into one account
- Fix a bill with an invalid rate
- ... (the list is only limited by your time and imagination)

Users execute these scripts via the business process assistant (BPA). Users can also define their favorite BPA scripts in their user preferences. By doing this, a user can execute a script by pressing an accelerator key (`Ctrl + Shift` + a number).

Don't think of these scripts as merely a training tool. BPA scripts can also reduce the time it takes to perform common tasks. For example, you can write a script that reduces the "number of clicks" required to add a new person to an existing account.

CAUTION: Future upgrade issues. Although we make every effort not to remove fields or tab pages between releases, there may be times when changes made by the base-package will necessitate changes to your scripts. Please refer to the release notes for a list of any removed fields or tab pages.

CAUTION: Scripts are not a substitute for end-user training. Scripts minimize the steps required to perform common tasks. Unusual problems (e.g., a missing meter exchange) may be difficult to script as there are many different ways to resolve such a problem. However, scripts can point a user in the right direction and reduce the need to memorize obscure business processes.

The topics in this section describe background topics relevant to BPA scripts.

How To Invoke Scripts

Refer to Initiating Scripts for a description of how end-users initiate scripts.

Developing and Debugging Your BPA Scripts

We recommend considering the approaches outlined below when you construct scripts.

While designing your scripts, determine the most maintainable way to set them up. Rather than creating complex, monolithic scripts, we recommend dividing scripts into smaller sections. For example

- Determine if several scripts have similar steps. If so, set up a script that contains these common steps and invoke it from the main scripts using a **Perform script** step.
- Determine if a large script can be divided into logical sections. If so, set up a small script for each section and create a "master script" to invoke each sub script via a **Transfer control** step.

For debugging purposes, you may find it helpful to categorize the step types into two groups: those that involve some type of activity in the script area, and those that don't. The following step types cause activity in the script area: **Height**, **Display text**, **Prompt user**, **Input data**, **Input Map**, **Set focus to a field**.

The rest of the step types are procedural and involve no user interaction. There are two techniques you can use to assist in debugging these step types.

- You can instruct the system to display text in the script area.
- You can display an entire data area (or a portion thereof) in the script area by entering **%+...+%** where **...** is the name of the node whose element(s) should be displayed.

Copyright © 2024, Oracle and/or its affiliates.

> **Note: Time saver.** When you develop a new BPA script, change your user preferences to include the script as your first "favorite". This way, you can press `Ctrl+Shift+1` to invoke the script (eliminating the need to select it from the script menu).

Launching A Script From A Menu

You can create menu items that launch BPA scripts rather than open a page. To do this, create a navigation option that references your script and then add a menu item that references the navigation option.

If the navigation option is referenced on a context menu and the navigation option has a "context field", a temporary storage variable will be created and populated with the unique identifier of the object in context. For example, if you add a "script" navigation option to the bill context menu and this navigation option has a context field of BILL_ID, the system will create a temporary storage variable called BILL_ID and populate it with the respective bill id when the menu item is selected.

Launching A Script When Starting The System

You can set the system to launch a script upon startup. This may be used for integrations from an external system or for use cases within the application where a link has been configured to launch the application in a new browser tab, allowing the original page to remain open. Refer to Creating Application Links in External Applications for details of what is supported.

Executing A Script When A To Do Entry Is Selected

The system creates To Do entries to highlight tasks that require attention (e.g., records in error). Users can complete many of these tasks without assistance. However, you can set up the system to automatically launch a script when a user selects a To Do entry. For example, consider a To Do entry that highlights a bill that's in error due to an invalid mailing address. You can set up the system to execute a script when this To Do entry is selected by a user. This script might prompt the user to first correct the customer's default mailing address and then re-complete the bill.

The following points provide background information to help you understand how to implement this functionality:

- Every To Do entry references a To Do type and a message category and number. The To Do type defines the category of the task (e.g., bill errors). The message number defines the specific issue (e.g., a valid address can't be found.). Refer to The Big Picture of System Messages for more information about message categories and numbers.
- When a user drills down on a To Do entry, either a script launches OR the user is transferred to the transaction associated with the entry's To Do type. You control what happens by configuring the To Do type accordingly:
  - If you want to launch a script when a user drills down on an entry, you link the script to the To Do type and message number. Keep in mind that you can define a different script for every message (and some To Do types have many different messages).
  - If the system doesn't find a script for an entry's To Do type and message number, it transfers the user to the To Do type's default transaction.

> **Note: How do you find the message numbers?** We do not supply documentation of every To Do type's message numbers (this list would be impossible to maintain and therefore untrustworthy). The best way to determine which message numbers warrant a script is during pre-production when you're testing the system. During this period, To Do entries will be generated. For those entries that warrant a script, simply display the entry on To Do maintenance. On this page, you will find the entry's message number adjacent to the description.

- These types of scripts invariably need to access data that resides on the selected To Do entry. Refer to How To Use To Do Fields for the details.

The Big Picture Of Script Eligibility Rules

You can configure eligibility criteria on the scripts to limit the scripts that a user sees in the script search. For example, you could indicate a script should only appear on the script menu if the user belongs to the level 1 customer service representative group. You may also indicate that a script should only appear if the data a user is viewing has certain criteria. For example, if you are using Oracle Utilities Customer Care and Billing, you can indicate that a script should

Copyright © 2024, Oracle and/or its affiliates.

only appear if the current account's customer class is residential. By doing this, you avoid presenting the user with scripts that aren't applicable to the current data in context or the user's role.

The topics in this section describe eligibility rules.

Script Eligibility Rules Are Not Strictly Enforced

The script search gives a user a choice of seeing all scripts or only scripts that are eligible (given the current data in context and their user profile). This means that it's possible for a script that isn't eligible for the given context data / user to be executed via this search. In other words, the system does not strictly enforce a script's eligibility rules.

It might be more helpful to think of eligibility rules as "highlight conditions". These "highlight conditions" simply control whether the script appears in the script search when a user indicates they only want to see eligible scripts.

You Can Mark A Script As Always Eligible

If you don't want to configure eligibility rules, you don't have to. Simply indicate that the script is always eligible.

You Can Mark A Script As Never Eligible

If you have scripts that you do not want a user to select from the script menu, indicate that it is never eligible. An example of a script that you wouldn't want a user to select from the menu is one that is launched when a To Do entry is selected. These types of scripts most likely rely on data linked to the selected To Do entry. As a result, a user should only launch scripts of this type from the To Do entry and not from the script menu.

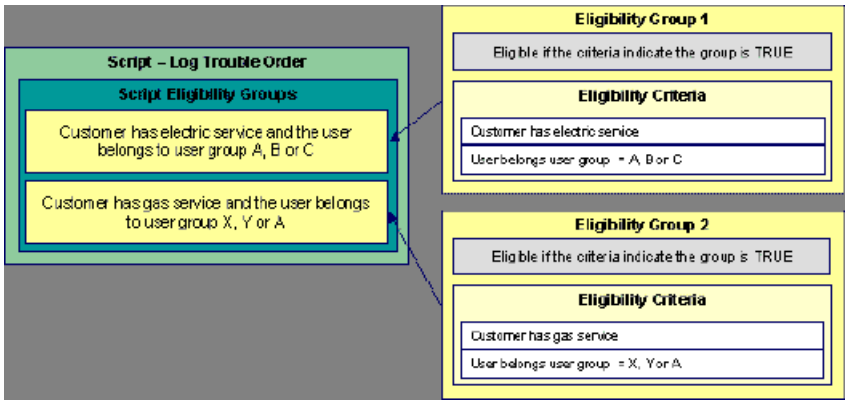
Criteria Groups versus Eligibility Criteria

Before we provide concrete examples of eligibility criteria, we need to explain two concepts: Criteria Groups and Eligibility Criteria. A script's criteria groups control whether a user is eligible to choose a script. At a high level, it works like this:

- A criteria group has one or more eligibility criteria. A group's criteria control whether the group is considered true or false.
- When you create a group, you define what should happen if the group is true or false. You have the following choices:
  - The user is eligible to choose the script.
  - The user is not eligible to choose the script.
  - The next group should be checked.

We will use the following example from Oracle Utilities Customer Care and Billing to help illustrate these points. Assume a script is only eligible if:

- The customer has electric service and the user belongs to user group A, B or C.
- OR, the customer has gas service and the user belongs to user group X, Y or A.



This script requires two eligibility groups because it has two distinct conditions:

- IF (Customer has electric service AND (User belongs to user group A, B or C))
- IF (Customer has gas service AND (User belongs to user group X, Y or A))

Copyright © 2024, Oracle and/or its affiliates.

If either condition is true, the script is eligible.

You would need to set the following criteria groups in order to support this requirement:

Group No.	Group Description	If Group is True	If Group is False
1	Customer has electric service and the user belongs to user group A, B or C	**Eligible**	**Check next group**
2	Customer has gas service and the user belongs to user group X, Y or A	**Eligible**	**Ineligible**

The following criteria are required for each of the above groups:

Group 1: Customer has electric service and the user belongs to user group A, B or C				
Seq	Logical Criteria	If Eligibility Criteria is True	If Eligibility Criteria is False	If Insufficient Data
10	Customer has electric service	**Check next condition**	**Group is false**	**Group is false**
20	User belongs to user group A, B or C	**Group is true**	**Group is false**	**Group is false**

Group 2: Customer has gas service and the user belongs to user group X, Y or A				
Seq	Logical Criteria	If Eligibility Criteria is True	If Eligibility Criteria is False	If Insufficient Data
10	Customer has gas service	**Check next condition**	**Group is false**	**Group is false**
20	User belongs to user group X, Y or A	**Group is true**	**Group is false**	**Group is false**

The next section describes how you might configure the specific logical criteria in each of the groups.

Defining Logical Criteria

When you set up an eligibility criterion, you must define two things:

- The field to be compared
- The comparison method

You have the following choices in respect of identifying the *field to be compared* :

- You can execute an algorithm to retrieve a field value from somewhere else in the system.
- Some products may support choosing a characteristic linked to an appropriate object in the system (such as an account or person).

You have the following choices in respect of identifying the *comparison method*:

- You can choose an operator (e.g., >, <, =, BETWEEN, IN, etc.) and a comparison value.
- You can execute an algorithm that performs the comparison (and returns True, False or Insufficient Data). This is also a very powerful feature, but it's not terribly intuitive. We'll present a few examples later in this section to illustrate the power of this approach.

The Examples Of Script Eligibility Rules provide examples to help you understand this design.

Copyright © 2024, Oracle and/or its affiliates.

Examples Of Script Eligibility Rules

The topics in this section provide examples about how to set up script eligibility rules.

A Script With A Time Span Comparison

A script that is only eligible for senior citizens has the following eligibility rules:

- Customer class = Residential
- Birth date equates to that of a senior citizen

These rules require only one eligibility group on the script. It would look as follows:

Group No.	Group Description	If Group is True	If Group is False
1	Residential and Senior Citizen	**Eligible**	**Ineligible**

The following criteria will be required for this group:

Group 1: Residential, Calif, Senior					
Seq	**Field to Compare**	**Comparison Method**	**If True**	**If False**	**If Insufficient Data**
10	Algorithm: retrieve account's customer class	= R	**Check next condition**	**Group is false**	**Group is false**
30	Person characteristic: Date of Birth	Algorithm: True if senior	**Group is true**	**Group is false**	**Group is false**

The first criterion is easy; it calls an algorithm that retrieves a field on the current account. This value, in turn, is compared to a given value. If the comparison results in a True value, the next condition is checked. If the comparison doesn't result in a True value, the **Group is false** (and, the group indicates that if the group is false, the script isn't eligible).

The last criterion contains a time span comparison. Time span comparisons are used to compare a date to something. In our example, we have to determine the age of the customer based on their birth date. If the resultant age is > 65, they are considered a senior citizen. To pull this off, you can take advantage of a comparison algorithm supplied with the base script as described below.

- Field to Compare. The person characteristic in which the customer's birth date is held is selected.
- Comparison Method. We chose a comparison algorithm that returns a value of **True** if the related field value (the customer's date of birth) is greater than 65 years (refer to **SECC-TIMESPN** for an example of this type of algorithm). Note that this time span algorithm can be used to compare days, weeks and months as well.

You'll notice that if a value of **True** is returned by the **True if senior** algorithm, the group is true (and we've set up the group to indicate a true group means the script is eligible).

A Script With Service Type Comparison

Imagine a script that is only eligible if the current customer has gas service and the user belongs to user groups A, B or C. This script would need the following eligibility rules:

- Customer has gas service
- User belongs to user group A, B, or C

These rules require only one eligibility group on the script. It would looks as follows:

Group No.	Group Description	If Group is True	If Group is False

Copyright © 2024, Oracle and/or its affiliates.

1	Has gas service and user is part of user group A, B or C	**Eligible**	**Ineligible**

The following criteria are required for this group:

Group 1: Has gas service and user is part of user group A, B, or C					
Seq	Field to Compare	Comparison Method	If True	If False	If Insufficient Data
10	Algorithm: check if customer has gas service	= True	**Check next condition**	**Group is false**	**Group is false**
20	Algorithm: check if user belongs to user group A, B or C	= True	**Group is true**	**Group is false**	**Group is false**

Both criteria are similar - they call an algorithm that performs a logical comparison. These algorithms are a bit counter intuitive (but understanding them provides you with another way to implement complex eligibility criteria):

The first criterion works as follows:

- Field to Compare. We chose a "field to compare" algorithm that checks if the current account has service agreements that belong to a given set of service types. It returns a value of **True** if the customer has an active service agreement that matches one of the service types in the algorithm. In our example, the "check if customer has gas service" algorithm returns a value of **True** if the customer has at least one active service agreement whose SA type references the gas service type. The "check if customer has electric service" algorithm is almost identical, only the service type differs.
- Comparison Method. We simply compare the value returned by the algorithm to True and indicate the appropriate response.

The second criterion works similarly:

- Field to Compare. We chose a "field to compare" algorithm that checks if the user belongs to any user group in a set of user groups. It returns a value of **True** if the user belongs to at least one user group defined in parameters of the algorithm. Refer to algorithm type **SECF-USRNGRP** for an example of this type of logic.
- Comparison Method. We simply compare the value returned by the algorithm to True and indicate the appropriate response.

Note: Bottom line. The "field to compare" algorithm isn't actually returning a specific field's value. Rather, it's returning a value of **True** or **False**. This value is in turn, compared by the "comparison method" and the group is set to true, false or check next accordingly.

The Big Picture Of Server-Based Scripts

Fastpath: Refer to The Big Picture Of Scripts to better understand the basic concept of scripts.

Server-based scripts allow an implementation to configure backend business processes. The system supports three types of server-based scripts, **Plug-In** scripts, **Service** scripts and **Groovy Library** scripts.

- Plug-in scripts allow an implementation to develop routines that are executed from the system's various plug-in spots. For example, an implementation could configure a plug-in script that is executed every time an adjustment of a given type is frozen.
- Service scripts allow an implementation to develop common routines that are invoked from both front-end and back-end services. For example, an implementation could create a service script that terminates an account's automatic payment preferences. This service script could be invoked from a BPA script initiated by an end-user when a customer

Copyright © 2024, Oracle and/or its affiliates.

asks to stop paying automatically, and it could also be executed from a plug-in script if a customer fails to pay their debt on time. Service scripts are typically written using xpath scripting.

- Groovy Library scripts allow an implementation to develop groups of common routines written in the Groovy language which may be invoked from **Groovy Member** step types.

The topics in this section describe background topics relevant to server-based scripts.

Additional Coding Options For Server Scripts

Server based scripts often perform complex functions best supported by coding in languages with more comprehensive command sets than the base script steps. The system supports two common third-party languages for this purpose.

XML Path Language (XPath) is a language for querying and evaluating elements or nodes in an XML document. XPath commands and expressions can be used directly within edit data step types. The script engine version is used to define the applicable XPath version.

Groovy is an object-oriented, dynamic language for the Java platform. The framework supports the use of Groovy within server-based scripts to provide restricted and controlled access to Java-like code, particularly for cloud based implementations. The following topic provides more information on how to incorporate Groovy code into scripts.

Using Groovy Within Scripts

Groovy code can be incorporated in scripts using the step type **Groovy Members**. For each script with Groovy code, there will be a single Groovy class created by concatenating all **Groovy Members** steps.

For security, the product and third party Java classes available for scripting in Groovy are restricted. The allowable base classes may be viewed in the Groovy Javadocs viewer via the 'View Groovy Javadocs' link in the context sensitive Script Tips zone. The list of allowable third party classes can be viewed via the 'View third party Groovy allowlist' link in the Script Tips zone.

Note: This system supports the use of Groovy for back end processing purposes. It is not intended for user interfaces. Groovy is also not applicable to BPA scripts.

Groovy Script Engine

For scripts invoked from within Java and entirely written in Groovy should use a script engine version of **Groovy**. The system processes this type of script in a similar way to code written in Java. This avoids the need to convert the script to and from XML format and allows the use of code that acts directly on the system objects with consequent performance benefits.

The following script types support the Groovy engine version:

Plug In Scripts

Plug-in Scripts can be configured to use the Groovy engine if they contain only Groovy oriented step types. The system provides an automatically generated superclass that defines the plug-in spot API. Internally, the Groovy code must conform to the system conventions for Java based algorithm types, including the inclusion of an 'invoke' method that is the plug-in entry point, and the definition of 'soft' parameters using annotations.

Groovy Library Scripts

Groovy Library Scripts provide the ability to create groups of common routines written in Groovy that can be called from within other scripts. Scripts of this type must include a single step type of **Groovy Library Interface** in which the publicly available methods in the library are listed. The supporting code for those methods is defined in one or more **Groovy Members** step types within the library script. The methods defined in the library can accept arguments and return values of any type. Scripts of this type use the Groovy engine by default and cannot include scripting step types.

Scripts that need to invoke methods from a Groovy library can use the createLibraryScript method provided by the system to instantiate the library interface. Ensure that the class being instantiated includes the .class filename extension.

Copyright © 2024, Oracle and/or its affiliates.

Invoking Groovy within the Scripting Engine

If a plug-in script or a service script is configured to use a scripting engine version, it can include a mixture of regular and **Groovy Members** step types. The script step types will define the process to be executed. The **Groovy Members** steps contain code that can be called from **Edit Data** step types within the script using the **invokeGroovy** command. Only Groovy methods that receive no arguments and return void are supported using this command. Refer to the section on edit data steps for more details.

For scripts using this option, the framework provides a superclass containing methods that support common scripting actions such as move commands, string evaluation, and methods to invoke business objects, business services and service scripts. Refer to the Groovy specific Javadocs for details of the supported methods.

Importing Groovy Classes

All the Groovy classes used by your script should be explicitly declared in a **Groovy Imports** step type. When saving a script, if the system finds classes referenced in the script that are not explicitly declared, the system issues a warning and automatically adds import statements for the classes. Script writers should review the imports to confirm that they are the correct classes (in case there is a class with the same name in two different packages).

Backward Compatibility

In previous releases, the system did not require explicit imports and had logic to automatically import all callable classes available to Groovy at run-time. For any existing script that does not explicitly declare its imports, the system continues to fall back on importing all callable classes at runtime. However, the recommendation is to identify and update any such script as this is not the recommended practice.

The system provides a batch program **F1-CAGVY** that reports all scripts that required the automatic import step to successfully compile. In addition, this batch program is useful when the version of Groovy is upgrade to proactively detect possible issues in compilation of existing scripts that may occur due to changes in Groovy. Implementations are encouraged to run this batch program in the short term to identify and fix any scripts missing explicit imports and also for any Groovy version upgrade to review any implementation owned scripts that may be reported.

Plug-In Scripts

Note: This section assumes you are familiar with the notion of plug-in spots (algorithm entities) and plug-ins. See The Big Picture Of Algorithms for more information.

As an alternative to writing a java program for a plug-in spot, the framework supports creating plug-ins using the defined script steps, Xpath commands, Groovy code or a combination of these three options.

The following topics describe basic concepts related to plug-in scripts.

A Plug-In Script's API

Like program-based plug-ins, plug-in scripts:

- Run on the application server.
- Have their API (input / output interface) defined by the plug-in spot (i.e., plug-in scripts do not get to declare their own API).
- Can only be invoked by the "plug-in spot driver".

For plug-ins configured to use a script engine version, the best way to understand the script's API is to use the **View Script Schema** hyperlink to view its parameters data area schema.

Copyright © 2024, Oracle and/or its affiliates.

```
<schema>
  <parm type="group">
    <soft type="list">
      <value/>
    </soft>
    <hard type="group">
      <action use="input"/>
      <businessObject type="group" use="input">
        <id/>
      </businessObject>
```

Notice the two groups: soft and hard. If you are familiar with plug-in spots, you'll recognize these as the classic soft and hard parameters:

- The **soft** parameters are the values of the parameters defined on the algorithm. Notice they are not named - if you want to reference them in your plug-in script, you must do it by position.
- The **hard** parameters are controlled by the plug-in spot (i.e., the algorithm entity). Notice that this plug-in spot has a single input parameter called " **businessObject/id"**. Also notice the **use=** attribute - this shows that this parameter is input-only (i.e., you can't change it in the plug-in script).

Note: XPath. You can click on an element name to see the XPath used to reference the element in your script.

Plug-ins configured to use the **Groovy** engine version do not use an XML interface for the API and instead are processed in the same way as Java algorithms. The framework supplies a dynamically generated superclass that implements the plug-in spot for Groovy objects. Use the **View Script Superclass** hyperlink to view this superclass and the methods to set and get the hard parameters.

Note: For plug-in scripts using the **Groovy** engine version, soft parameters do not appear in the plug-in spot API as defined by the superclass. Plug-ins using only Groovy code define their soft parameters using annotations, in a similar way to Java algorithms, and fetch those values using methods defined in the algorithm code.

Setting Up Plug-In Scripts

The following points describe how to implement a plug-in script:

- Compose your plug-in script, associating it with the appropriate algorithm entity (plug-in spot).
- Create a new algorithm type for the respective algorithm entity, referencing your plug-in script as the program to carry out the algorithm type's function. Only plug-in scripts associated with the algorithm entity may be referenced on the algorithm type.
- Set up an algorithm for the respective algorithm type and plug it in where applicable. Refer to Setting Up Algorithm Types for more information.

Service Scripts

BPA scripts run on the client's browser and guide the end-users through business processes. Service scripts run on the application server and perform server-based processing for BPA scripts, zones and more. You may want to think of a service script as a common routine that is set up via scripting (rather than programming).

The following topics describe basic concepts related to service scripts.

A Service Script's API

As with any common routine, a service script must declare its input / output parameters (i.e., its API). A service script's API is defined on its schema.

Copyright © 2024, Oracle and/or its affiliates.

> **Note:** Refer to Schema Nodes and Attributes for a complete list of the XML nodes and attributes available to you when you construct a schema.

Invoking Service Scripts

Any type of script configured to use a scripting engine version may invoke a service script:

- A BPA script may invoke a service script to perform server-based processing.
- Plug-in scripts may invoke a service script (like a "common routine").
- A service script may call another service script (like a "common routine").

Map zones may be configured to invoke service scripts to obtain the data to be displayed. Refer to Map Zones for more information.

Inbound web services support interaction with service scripts allowing the outside world to interact directly with a service script.

You can also invoke a service script from a Java class.

Groovy Library Scripts

Just as service scripts can define common routines written in scripting language, Groovy library scripts are used to define groups of common components or methods written in Groovy. Groovy library code runs on the application server and performs server-based processing for scripts that utilize Groovy code.

The following topics describe basic concepts related to Groovy library scripts.

A Groovy Library Script's API

A Groovy library script's API is composed of one or more public methods whose code is defined in the script's steps. Those methods are defined in a step type of **Groovy Library Interface**. A Groovy library script must have one (and only one) step of this type.

Invoking Groovy Library Methods

Any type of script that supports the **Groovy Members** step type may invoke common methods defined in Groovy library scripts.

Code within a **Groovy Members** step must create an instance of the Groovy library interface definition to enable the interface methods to be invoked. Refer to the topic Using Groovy Within Scripts for more information.

Debugging Server-Based Scripts

The server can create log entries to help you debug your server scripts.

The logs contain a great deal of information including the contents of the referenced data area for **Move data**, **Invoke business object**, **Invoke business service** and **Invoke service script** steps.

Refer to the Debug Mode topic in the Configuration Tools chapter for details of how to execute the application in debug mode.

Maintaining Scripts

The script maintenance transaction is used to maintain your scripts. The topics in this section describe how to use this transaction.

> **Fastpath:** Refer to The Big Picture Of Scripts for more information about scripts.

Script - Main

Use this page to define basic information about a script. Open this page using **Admin** > **System** > **Script**.

Copyright © 2024, Oracle and/or its affiliates.

Use the **Script Query** portal to search for an existing script. Once a script is selected, you are brought to the maintenance portal to view and maintain the selected record.

Note: Script Tips. A context sensitive "Script Tips" zone is associated with this page. The zone provides links to Edit Data Syntax and Advanced Schema Topics so that users can quickly access those online help topics to aid in constructing scripts. In addition, the zone provides links to view the Groovy Javadocs viewer and the allowlist of third party Groovy classes so that users can verify the restricted list of classes available for Groovy coding in the script.

Description of Page

Enter a unique **Script** code and **Description** for the script. Use the **Detailed Description** to describe the purpose of this script in detail. **Owner** indicates if the script is owned by the base package or by your implementation (**Customer Modification**).

CAUTION: Important! If you introduce a new script, carefully consider its naming convention. Refer to System Data Naming Convention for more information.

Script Type indicates if this is a **BPA Script**, **Plug-In Script**, **Groovy Library Script** or **Service Script**. Refer to The Big Picture Of BPA Scripts and The Big Picture Of Server Based Scripts for more information.

Accessibility Option appears only for BPA scripts. Set this value to **Accessible from Script Menu** for any script that may be launched as a stand-alone script. Scripts with this configuration may be linked to a navigation option so that they may be invoked from a menu and may be configured by a user as a favorite script. Set this value to **Not Accessible from Script Menu** for any script that cannot be launched on its own. For example, any script that is a invoked as a sub-script from another script should have this setting. In addition, any script that is designed to be launched from within a specific portal where certain data is provided to the script should include this setting.

Enter an **Application Service** if the execution of the script should be secured. The application service should include **Execute** as one of its access modes. Refer to Securing Script Execution for more information. This field does not appear if the script type is **Groovy Library Script**.

Algorithm Entity appears only for plug-in scripts. Use this field to define the algorithm entity into which this script can be plugged in.

Business Object appears only for business object related plug-in scripts. Enter the Business Object whose elements are to be referenced by the plug-in script.

Script Engine Version defines key information affecting the context and execution of the script.

- Script engine version values of 1, 2 and 3 define the version of the XML Path Language (XPath) to be used for the script. Versions 2 and 3 use the XPath 2 engine supplied by the XQuery team. This is the same engine used inside the Oracle database. The current script engine version 3 is a modified version that offers performance improvements without impacting existing version 2 scripts.

  The default script engine version is 3.0 for plug-in and service scripts. The default version for BPA scripts is 1.0 as higher level versions are not applicable.

  There are some additional details to note about script engine version 1.0:

  - The XPath library used is Jaxen
  - For BPA scripts, it uses the browser's xpath and XML support except for Internet Explorer where the XSXML parser is used.
  - Xpath 1 (and even JavaScript) uses floating point arithmetic, which means that adding a collection of numbers with two decimal places might end up with a value of 10779.079999999998 instead of 10779.08
- A Script Engine Version value of **Groovy** is only applicable to plug-in scripts. It is used to indicate that only Groovy related step types are used in the script and signals to the system that there is no need to convert the data to and from an XML interface. This allows for greater efficiency in script execution.
- The value **Framework Version 2.1 Compatibility Mode** remains for upgrade purposes. This value should only be applicable to early versions of BPA scripts using syntax that is incompatible with XPath syntax.

Copyright © 2024, Oracle and/or its affiliates.

Note: The **Script Engine Version** field does not appear for **Groovy Library** scripts. The script engine version for these scripts is set to **Groovy** by default and cannot be changed.

Click the **View Script Schema** to view the script's data areas on the schema viewer window. This link does not appear if the script engine version is **Groovy**.

Click the **View XSD** hyperlink to view a script's schema definition in XSD format. This link only appears if the script type is **BPA Script** or **Service Script**.

The **View Script Superclass** hyperlink appears only for plug-in scripts using an engine version of **Groovy**. Click this link to view the code of the runtime generated superclass for the related plug-in spot's implementation.

The **View Script As Text** hyperlink appears for server-based scripts only. Click this link to view the internal scripting commands in a separate window.

The tree summarizes the script's steps. You can use the hyperlink to transfer to the **Step** tab with the corresponding step displayed.

Script - Step

Use this page to add or update a script's steps. Open this page using **Admin** > **System** > **Script** and then navigate to the **Step** tab.

Note: Time saver. You can navigate to a step by clicking on the respective node in the tree on the Main tab.

Description of Page

The **Steps** accordion contains an entry for every step linked to the script. When a script is initially displayed, its steps are collapsed. To see a step's details, simply click on the step's summary bar. You can re-click the bar to collapse the step's details. Please see accordions for the details of other features you can use to save time.

Select the **Step Type** that corresponds with the step. Refer to How To Set Up Each Step Type for an overview of the step types.

CAUTION: The Step Type affects what you can enter on other parts of this page. The remainder of this section is devoted to those fields that can be entered regardless of Step Type. The subtopics that follow describe those fields whose entry is contingent on the Step Type.

Step Sequence defines the relative position of this step in respect of the other steps. The position is important because it defines the order in which the step is executed. You should only change a Step Sequence if you need to reposition this step. But take care; if you change the Step Sequence and the step is referenced on other steps, you'll have to change all of the referencing steps.

Note: Leave gaps in the sequence numbers. Make sure you leave space between sequence numbers so that you can add new steps between existing ones in the future. If you run out of space, you can use the **Renumber** button to renumber all of the steps. This will renumber the script's steps by 10 and change all related references accordingly.

Display Step is only enabled on BPA scripts for step types that typically don't cause information to be displayed in the script area (i.e., step types like **Conditional Branch**, **Go to a step**, **Height**, etc). If you turn on this switch, information about the step is displayed in the script area to help you debug the script.

CAUTION: Remember to turn this switch off when you're ready to let users use this script.

Note: If **Display Step** is turned on and the step has **Text**, this information will be displayed in the script area. If **Display Step** is turned on and the step does not have **Text**, a system-generated messages describing what the step does is displayed in the script area.

Display Icon controls the icon that prefixes the **Text** that's displayed in the script area. Using an icon on a step is optional. This field is only applicable to BPA scripts.

Copyright © 2024, Oracle and/or its affiliates.

Text is the information that displays in the script area when the step executes. You need only define text for steps that cause something to display in the script area.

Fastpath: Refer to How To Substitute Variables In Text for a discussion about how to substitute variables in a text string.

Fastpath: Refer to How To Use HTML Tags And Spans In Text for a discussion about how to format (with colors and fonts) the text that appears in the script area.

The other fields on this page are dependent on the **Step Type**. The topics that follow briefly describe each step type's fields and provide additional information about steps.

Click on the **View Script Schema** hyperlink to view the script's data areas. Doing this opens the schema viewer window.

The **View Script As Text** hyperlink appears for server-based scripts only. Click this link to view the internal scripting commands in a separate window. The presented script syntax is valid within edit data steps.

How To Set Up Each Step Type

The contents of this section describe how to set up each type of step. Note that some step types are only applicable for BPA scripts and some step types are only supported for server based scripts. Each section indicates whether there are any script type limitations.

Conditional Branch Steps

Conditional branch steps allow you to conditionally jump to a different step based on logical criteria. For example, you could jump to a different step in a script if the customer is residential as opposed to commercial. In addition, several fields are required for **Conditional Branch** steps:

Compare Field Type and **Compare Field Name** define the first operand in the comparison. The **Field Type** defines where the field is located. The **Field Name** defines the name of the field. The following points describe each field type:

- **Current To Do Information**. Use this field type when the field being compared resides on the current To Do entry. Refer to How To Use To Do Fields for instructions on how to define the appropriate **Field Name**.
- **Data Area**. Use this field type when the field being compared is one that you put into one of the scripts data areas in an earlier step. **Field Name** must reference both a data area structure name as well as the field, for example "parm/charType". Refer to How To Reference Fields In Data Areas for instructions on how to construct the appropriate **Field Name**.
- **Page Data Model**. Use this field type when the field being compared resides on one of the tab pages in the object display area. Refer to How To Find The Name Of Page Data Model Fields for instructions on how to find the appropriate **Field Name**.
- **Predefined Value**. Use this field type when the field being compared is a global variable.
- **Temporary Storage**. Use this field type when the field being compared is one that you put into temporary storage in an earlier step. The **Field Name** must be the same as defined in an earlier step.
- **User Interface Field**. Use this field type when the field being compared resides on the currently displayed tab page. Refer to How To Find The Name Of User Interface Fields for instructions on how to find the appropriate **Field Name**.

Condition defines the comparison criteria:

- Use >, <, =, >=, <=, <> (not equal) to compare the field using standard logical operators. Enter the comparison value using the following fields.
- Use **In** to compare the first field to a list of values. Each value is separated by a comma. For example, if a field value must equal **1**, **3** or **9**, you would enter a comparison value of **1,3,9**.
- Use **Between** to compare the field to a range of values. For example, if a field value must be between **1** and **9**, you would enter a comparison value of **1,9**. Note, the comparison is inclusive of the low and high values.

Comparison Field Type, **Comparison Field Name** and **Comparison Value** define what you're comparing the first operand to. The following points describe each field type:

- **Current To Do Information**. Use this field type when the comparison value resides on the current To Do entry. Refer to How To Use To Do Fields for instructions on how to define the appropriate **Field Name**.

Copyright © 2024, Oracle and/or its affiliates.

- **Data Area**. Use this field type when the comparison value resides in one of the scripts data areas. **Field Name** must reference both a data area structure name as well as the field, for example "parm/charType". Refer to How To Reference Fields In Data Areas for instructions on how to construct the appropriate **Field Name**.
- **Page Data Model**. Use this field type when the comparison value resides on one of the tab pages in the object display area. Refer to How To Find The Name Of Page Data Model Fields for instructions on how to find the appropriate **Field Name**.
- **Predefined Value**. Use this field type when the field being compared is a constant value defined in the script. When this field type is used, use **Comparison Value** to define the constant value. Refer to How To Use Constants In Scripts for instructions on how to use constants.
- **Temporary Storage**. Use this field type when the comparison value is a field that you put into temporary storage in an earlier step. The **Field Name** must be the same as defined in an earlier step.
- **User Interface Field**. Use this field type when the comparison value resides on the currently displayed tab page. Refer to How To Find The Name Of User Interface Fields for instructions on how to find the appropriate **Field Name**.

Note: Conditional field types. The field types **Current To Do Information**, **Page Data Model** and **User Interface Field** are only applicable to BPA scripts. Only the field types **Temporary Storage** and **Predefined Value** are applicable when the Condition is **In** or **Between**.

The above fields allow you to perform a comparison that results in a value of **TRUE** or **FALSE**. The remaining fields control the step to which control is passed given the value:

- **If True, Go to** defines the step that is executed if the comparison results in a **TRUE** value.
- **If False, Go to** defines the step that is executed if the comparison results in a **FALSE** value.

Note: Numeric Comparison. Comparison of two values may be numeric or textual (left-to-right). Numeric comparison takes place only when values on both side of the comparison are recognized as numeric by the system. Otherwise, textual comparison is used. Fields for **Current To Do Information**, **Data Area**, **Page Data Model**, and **User Interface Field** types are explicitly associated with a data type and therefore can be recognized as numeric or not. This is not the case for fields residing in **Temporary Storage** or those set as **Predefined Values**. A **Temporary Storage** field is considered numeric if it either holds a numeric value moved to it from an explicitly defined numeric value (see above) or it is a resultant field of mathematical operation. A **Predefined Value** field is considered numeric if the other field it is compared to is numeric. For example, if a numeric field is compared to a **Predefined Value** the latter is considered numeric as well resulting in numeric value comparison. However, if the two fields are defined as **Predefined Values** the system assumes their values are text strings and therefore applies textual comparison.

Display Text Steps

Note: BPA scripts only. This step type is only applicable to BPA scripts.

Display text steps cause a text string to be displayed in the script area. Steps of this type can be used to provide the user with guidance when manual actions are necessary. In addition, they can be used to provide confirmation of the completion of tasks.

The information you enter in the **Text** field is displayed in the script area when the step is executed.

The text string can contain substitution variables and HTML formatting commands. Also note that for debugging purposes, you can display an entire data area (or a portion thereof) by entering **%+...+%** where **...** is the name of the node whose element(s) should be displayed.

Edit Data Steps

Edit data steps provide a free format region where you can specify commands to control your script processing.

Enter your scripting commands in the **Edit Data Text** field. Click the adjacent icon to open a window providing more space for defining the edit data step.

In general, the syntax available within edit data mimics the commands available within the explicit step types. However, there are a few commands that are available only within edit data. For example, the two structured commands: **For**, and **If**.

Copyright © 2024, Oracle and/or its affiliates.

Note: Not all BPA step types are supported using the edit data syntax. Refer to the details below for more information on edit data commands and examples.

The topics in this section provide detail of the syntax supported in the edit data step type.

Comments

You can place comments into your script using the double slash notation // in the first two characters of the edit data step. Example:

```
// -
// quit with error
//
if ("not(customer/securityEnabled)")
terminate with error (8000, 1001 %1="customer/id" %2='not allowed');
end-if;
```

Temporary Variables

Temporary variables can be declared within all types of scripts. They should be referenced by a leading single dollar sign ('**$**'). However, temporary variables behave differently in the various script types:

- In BPA Scripts temporary variables remain persistent from one BPA script to another (refer to the Perform Script and Transfer Control statements), which means that you can use temporary variables to communicate between BPA scripts.
- Temporary variables cannot be passed from a BPA script to a service or plug-in script. Only data area elements can be passed between these types of scripts.
- Within service and plug-in scripts, temporary variables remain persistent only for the life of the particular script that declared the variable. This means that temporary variables cannot be passed between plug-in scripts and service scripts, only global variables, context variables and data area elements can be passed between these types of scripts.

Declaring / Initializing / Defaulting Temporary Variables

When using a temporary variable, it should be declared or initialized with an appropriate value before using it. A typical method for declaring a variable is to simply move data to it in a move statement, for example.

```
move -"0" to $index;
```

Fastpath: Refer to Move to a Temporary Variable for more information on implicit declaration of a temporary variable within a move statement.

For BPA scripts, as mentioned above, temporary variables may be passed from one BPA script to another. As such, it is common to reference a temporary variable in a BPA that should have been initialized by a previous BPA. However, if there is any reason that a temporary variable did not get initialized by a previous BPA, a reference to it will cause an error. It is good practice, therefore, to use the **default** statement that will initialize temporary variables that are not already created / initialized.

- The following statement will initialize the temporary variable $InputAction – but only if the temporary variable has not yet been initialized:

  ```
  default $InputAction;
  ```

- The following statement will set the value of the temporary variable $InputAction to 'read' – but only if the variable has not yet been initialized:

  ```
  default $InputAction as -'read';
  ```

Note: Scripts should take care not to define variables using a reserved keyword. The following table lists the reserved keywords.

Copyright © 2024, Oracle and/or its affiliates.

Keyword
add
as
asError
bpa
branch
data
declareBO
declareBS
declareDA
declareMap
declareSS
default
delete
edit
element
else
end-edit
end-for
end-if
error
escape
evaluate
fastAdd
fastUpdate
for
goto
if
in
invokeBO
invokeBS
invokeMap
invokeSS
label
map

Copyright © 2024, Oracle and/or its affiliates.

Keyword
move
navigate
navigateAndReloadDashboard
null
page
performScript
popup
read
readWithoutVersion
replace
suppress
target
terminate
to
transferControl
update
using
warn
with

Context Variables

Context variables are only available within service scripts. The context variable will be available for the duration of the service script and all that it invokes. Therefore, you can use a context variable within a service script to communicate information to a lower level service script or schema. They should be referenced by leading double dollar signs ('**$$**').

Note: Because context variables are available for lower level scripts, they may sometimes be referred to as global variables or global context variables. But they should not be confused with global variables.

Declaring / Initializing / Defaulting Context Variables

When using a context variable, it should be declared or initialized with an appropriate value before using it. A typical method for declaring a variable is to simply move data to it in a move statement, for example.

```
move -'context variable' to $$contextVariable;
```

Fastpath: Refer to Move using a Context Variable for more information.

Copyright © 2024, Oracle and/or its affiliates.

Move Statement

The **move** statement copies a source value to a target. The following table highlights various options supported in the move statement.

Statement	Example Description	Example Syntax
Move to Element **move "xpath" to "xpath";** **Note:** An XPath expression is surrounded by double quotes.	Move statement with simple XPath reference.	`move -"acct/totalBalance" to -"parm/formattedValue";`
	Move statement with XPath concatenate function.	`move -"concat(person/firstName, -',', -person/lastName)" -to -"parm/fullName";`
	Move statement with XPath substring-before function.	`move -"substring-before(parm/fullName,',')" -to -"person/firstName";`
	Move statement with XPath substring-after function.	`move -"substring-after(parm/fullName,',')" -to -"person/lastName";`
	Move statement with XPath substring function.	`move -"substring(parm/date,1,4)" -to -"parm/year";`
Move to Element **move 'literal' to "xpath";** **Note:** A literal value is surrounded by single quotes.	Move statement using a literal string.	`move -'okay for -mailing' -to -"account/preferences[type="mail"]/text";`
Move to Element **move 'Boolean' to "xpath";**	Move statement with Boolean as literal string.	`if ("account/balance > 0") -  move -'true' -  to -"account/hasDebitBalance"; -end-if;`

	Moving an expression, which results in a Boolean. Note that the filter in the example below is located on a group node.	`<schema>  <account>  -<hasDebitBalance -type="boolean"/>  <balance/>  </account></schema>`
		
		`move -"boolean(account[balance>0]`

Copyright © 2024, Oracle and/or its affiliates.

Statement	Example Description	Example Syntax
		`to -"account/ hasDebitBalance";`

Move to Group
move "xpath" to "xpath";
	Move a set of elements from one group with another.
	The system matches the initial level of element names and the group / list names from the source schema to the target schema. For lists and groups, the default behavior is to move all elements within the source group / list to the target group / list (even if they are not defined in the target group / list).
	The business service F1-MoveByName may be used for more granular control of the move (without having to define each individual move statement).
		`move -"account/ custInfo" - to -"person";`
		This statement is equivalent to the following statement:
		`move -"account/ custInfo/*" - to -"person/*";`

Move using a Temporary Variable	When moving to a temporary local variable, it is not surrounded by double quotes. **move "xpath" to $variable;**	`move -"count(Person/ names/personName) - + count(Person/ids/ personId)" - to - $PersonChildCount;`
	When moving from a temporary variable, the variable is surrounded by double quotes. **move "$variable" to “xpath”;**	`move -"$AccountBalance" - to -"parm/ formattedValue";`
Move using a Context Variable **move "xpath" to $$variable;** **move $$variable to “xpath”;**	Context variables, source or target, are referenced without any double quotes.	`move -'context - value' - to - $$contextVariable; // - // here, we move - from a context - variable. move - $$contextVariable - to -"MarketMessage/ sender";`
Move using a Dynamic Location **move "xpath" | 'literal' to evaluate("xpath" | $variable);**	The evaluate statement allows your move source or target location to be dynamically derived from a variable or schema element location.	`move -'literal' - to - evaluate("schemaLocation"); - // move -"schemaLocation" -`

Copyright © 2024, Oracle and/or its affiliates.

Statement	Example Description	Example Syntax

move evaluate("xpath" | $variable) to "xpath" | $variable;		`to - evaluate($Variable);`
		
		`move - evaluate("schemaLocation") - to $Variable; // - move - evaluate($Variable) - to -"schemaLocation";`

Move escape **move escape("xpath" | $variable) to "xpath" | $variable;**	Move escape is only available for service scripts and plug-in scripts. The escape statement scans your source text value for HTML content and escapes it, i.e. replaces any HTML-like characters with special characters that are escaped from HTML rendering. By doing so the text would be displayed as plain text when displayed as part of an HTML element. **Note:** You should only use this function if the text is to be displayed as part of an HTML element and is suspected to contain HTML-like characters or even malicious HTML that should not be rendered as HTML. If incorrectly displayed using a non HTML element, the special escape characters, if any would be visible as part of your text. Refer to the UI Map Attributes and Functions for more information on how to define an element to display HTML content.	`move - escape("schemaLocation") - to $Variable; // - move - escape($Variable) - to -"schemaLocation";`
Move null **move null to "xpath";**	You can remove information from the XML instance document through the special syntax of move 'null'. Note that you can specify either a node name in the XPath expression or a group name. If you specify a group then the group and all child elements will be eliminated from processing.	Remove a node and all of its child nodes: `if - ("boolean(customer/securityEnabled)") goto updateInfo; else move null - to -"customer"; - end-if;`

Copyright © 2024, Oracle and/or its affiliates.

Statement	Example Description	Example Syntax

		Remove all child nodes of a group node with the suffix '/*'.
		`if -`
		`("boolean(customer/`
		`securityEnabled)")`
		`move null -`
		`to -"customer/*";`
		`end-if;`

Go To Statement

The edit data step supports functionality analogous to the Go To step type. The syntax is **goto label;** where the label represents another location within the edit data text field (identified by this label) or represents another step in the script.

The following is an example of going to another location in the same step identified by the label **addSpouse**.

```
if ("string(parm/spouse/name) -!= $BLANK")
  goto addSpouse;
end-if;
addSpouse: invokeBO -'Person' using -"parm/spouse" for add;
```

The following is an example of going to a different step within the same script. The step sequence is the reference used as the label.

```
if ("string(parm/spouse/name) -!= $BLANK")
  goto 110;
end-if;
.
.
.
110: invokeBO -'Person' using -"parm/spouse" for add;
```

Conditional Branch Statement

The edit data step supports functionality analogous to the Conditional Branch step type. The syntax is **branch (“xpath”) goto label else label;** where:

- The XPath condition in the **branch** statement must evaluate to a Boolean value of True or False.
- The targets for the **goto** and **else** statements are labels that represent another location within the edit data text field (identified by this label) or represent another step in the script.

The following example uses labels for **addSpouse** and **addAccount**

```
branch ("string(parm/spouse/name) -!= $BLANK") goto addSpouse else addAccount;
```

If Statement

The **if** statement is similar to the conditional branch statement. Either can be used to structure the logic of your script. This statement may optionally include an **else** statement but it should always end with an **end-if** statement.

Note: This is an example of a statement that is not represented as a separate step type. It is only available within the edit data text.

The syntax is **if (“xpath”) else end-if;**. The XPath condition must evaluate to a Boolean value of True or False. The following are some examples.

Copyright © 2024, Oracle and/or its affiliates.

Example where the XPath contains a simple logical condition.

```
if ("string(parm/spouse/name) -!= $BLANK")
  -//
  -// Create spouse since spouse name present
  goto addSpouse;
else
  -//
  -// Create account without spouse
  goto addAccount;
end-if;
```

Example where the XPath contains a complex condition.

```
if ("string(parm/spouse/name) -!= $BLANK and string(parm/hasSpouse) = true or -
boolean(parm/requireSpouse)")
  -//
  -// Create spouse since spouse name present
  goto addSpouse;
end-if;
```

Example of a stacked set of statements used to evaluate multiple possible values of a field.

```
if ("parm/rowCount = 0")
  -//
  -// no rows found
  goto quit;
end-if;
if ("parm/rowCount = 1")
  -//
  -// one row found
  goto process;
end-if;
if ("parm/rowCount > 1")
  -//
  -// more than one row found
  goto quit;
end-if;
quit: terminate;
```

The following XPath shows Boolean based on the existence of the node. In this example, if the node exists in the XML instance document being processed, the statement will evaluate to True. If no element is found, the statement evaluates to false.

Note: When treating XPath nodes as Boolean variables be aware that an empty node evaluates to True. Only a missing node return False.

```
if ("boolean(parm/spouse/name)")
  goto addSpouse;
else
  -//
  -// Create account without spouse
  goto addAccount;
end-if;
 -
if ("not(parm/spouse/name)")
  -//
  -// Create account without spouse
  goto addAccount;
else
  goto addSpouse;
end-if;
```

Copyright © 2024, Oracle and/or its affiliates.

For Statement

The **for** statement creates a list of nodes or values depending on your XPath expression. If you specify a list node then every child node of the list, along with its contents, will be available within the loop. If you specify a child node directly, then a list of values only will be available within the loop.

Note: For more information on creating new entries in a list, please refer to the creating a new list instance example.

Note: This is an example of a statement that is not represented as a separate step type. It is only available within the edit data text.

The syntax is **for ($variable in "xpathList") end-for;**. The XPath condition must evaluate to a Boolean value of True or False.

The following examples are based on this sample schema:

```
<schema>
  <SAList type="list">
    <id/>
    <balance/>
  </SAList>
  <SAContributor type="list">
    <id/>
  </SAContributor>
</schema>
```

Example that specifies the list node in the XPath expression where all child nodes are available for processing.

```
move -"0" to $AccountBalance;
move -"0" to $index;
for ($SAList in -"SAList")
    move -"$SAList/balance + $AccountBalance" to $AccountBalance;
    -//
    -// keep track of each SA contributing to the balance in the SA Contributor list
    move -"1 + $index" to $index;
    move -"$SAList/id" to -"SAContributor[$index]/id";
end-for; -
```

Example that specifies a child node within the list node in the XPath expression. Only values of that node are available for processing.

```
move -"0" to $AccountBalance;
for ($SABalance in -"SAList/balance")
    move -"$SABalance + $AccountBalance" to $AccountBalance;
end-for;
```

Example that shows that a filter can be used to limit the rows selected by the **for** loop.

```
move -"0" to $AccountDebitBalance;
for ($SAList in -"SAList[Balance>0]")
    move -"$SAList/balance + $AccountDebitBalance" to $AccountDebitBalance;
end-for; -
```

Example that shows the use of a filter when specifying child nodes.

```
move -"0" to $AccountCreditBalance;
for ($SABalance in -"SAList[Balance<0]/balance")                 -
    move -"$SABalance + $AccountCreditBalance" to -
$AccountCreditBalance;                -
end-for;
```

Copyright © 2024, Oracle and/or its affiliates.

List Processing

This section provides details about processing lists. The examples in this section reference the following schema:

```
<schema>
  <parm type="group">
    <name/>
  </parm>
  <Person type="group">
    <names type="list">
      <type/>
      <name/>
    </names>
  </Person>
</schema>
```

Referencing a List Element. You can move a value to a particular list instance by referencing an identifying node in the list within a filter. The syntax is **move "xpath" to "xpathList[filter]/element";** Example:

```
move -"parm/name" to -"Person/names[type='main']/name";
```

Creating a New List Instance. A special notation can be used within a move target statement to indicate a new list instance should be created. The "+" indicates to the script processor that a new instance of a list should be initiated for the target element. The syntax is **move "xpath" to "+xpathList";** Example:

```
move -"parm/name" to -"Person/+names/name";
```

Deleting a List Instance. An XML list entry can be deleted from the database by moving an action attribute of 'delete' to the element name. To cause a database delete of a list entry requires an attribute of action="delete" in the target node and a subsequent update BO interaction. The syntax is **move 'delete' to "xpathList@action");** Example:

```
if ("parm/action = -'d'") -
    move -"0" to $index;
    for ($CCList in -"CCList")
        move -"1 + $index" to $index;
        if ("$CCList/id = parm/id")
            move -'delete' to -"CCList[$index]@action";
            goto update;
        end-if;
    end-for;
end-if;
```

The following shows the resulting XML.

```
<root>
  <CCList>
    <id>9876538976</id>
    <balance>309.98</balance>
  </CCList>
  <CCList action="delete">
    <id>4321125899</id>
    <balance>87.45</balance>
  </CCList>
</root>
```

Note: Deleting a list instance through use of the action attribute is risky if iterative BO interactions are required. The XML document that contains the list instance to be deleted will not be altered after a successful BO interaction, which means the document will still contain the list instance even though it no longer exists. To solve this problem, it is essential to re-read the BO after any BO update where the action attribute of 'delete' has been used.

Copyright © 2024, Oracle and/or its affiliates.

Note: An alternative to the delete attribute described here, is to use the BO action of replace. Manipulating a list to use the replace action avoids the problem described above concerning stale information in request documents post BO update.

Functions for Processing a List

XPath provides several functions that are useful to process elements of a list including **count**, **sum** and **last**.

The following examples are based on this sample XML document:

```
<xml>
  <ft>
    <type>bill</type>
    <date>20100101</date>
    <amt>30.30</amt>
    <cat>tax</cat>
  </ft>
  <ft>
    <type>adj</type>
    <date>20100301</date>
    <amt>20.20</amt>
    <cat>int</cat>
  </ft>
  <ft>
    <type>bill</type>
    <date>20100201</date>
    <amt>10.10</amt>
    <cat>tax</cat>
  </ft>
</xml>
```

The following is an example of a sum. The syntax is **move "sum(xpathList/element)" to $variable;** The example sums the total balance.

```
move -"sum(ft/amt)" to $TotalBalance;
```

The following is an example of a sum using a filter to get a subtotal. The example sums the balance of the entries that have the ‘tax’ category.

```
move -"sum(ft[cat='tax']/amt)" to $TaxBalance;
```

The following is an example of a count. The syntax is **move "count(xpathList)" to $variable;** The example finds the count of the number of FT entries in the list.

```
move -"count(ft)" to $TranCount;
```

The following is an example of ‘last’, which is used to locate the last entry. The syntax is **move "last(xpathList)" to $variable;** The example finds the last amount in the FT list.

```
move -"ft[last()]/amt" to $LastAmount;
```

Declare and Invoke Schema Based Objects

You can invoke a business object, business service or service script within the edit data step. To support the dynamic invoke, a dynamic data area name can be declared.

The schema being declared may be a business object (BO) schema, a business service (BS) schema, a service script (SS) schema, data area (DA) schema or a UI map schema. The declare statement will differ based on the type of schema, but the syntax is analogous.

- **declareBO 'BO Name' | $variable | "xpath" as 'DynamicDataArea';**
- **declareBS 'BS Name' | $variable | "xpath" as 'DynamicDataArea';**

Copyright © 2024, Oracle and/or its affiliates.

- **declareSS 'SS Name' | $variable | "xpath" as 'DynamicDataArea';**
- **declareDA 'DA Name' | $variable | "xpath" as 'DynamicDataArea';**
- **declareMap 'Map Name' | $variable | "xpath" as 'DynamicDataArea';**

When invoking a BO, BS or SS, the name of the object can be specified as a literal or it can be a value contained within an element or a variable. For every Invoke, you must supply an XPath reference to a group name.

When invoking a business object, an action must be supplied. The syntax is **invokeBO 'BO Name' | $variable | "xpath" using "xpath" for action;**. The valid actions are as follows:

- **read**. This action reads the current view of the BO data.
- **add**. This action will add the object and read and return the resulting view of the BO.
- **fastAdd**. This action will add the object but does not perform a subsequent 'read' to return the resulting view of the BO. This option is better than add for performance purposes if there is no reason to re-read the record.
- **update**. This action will update the object and read and return the resulting view of the BO. This action executes a 'merge' of the information specified in the invoke statement's request XML document with existing BO data. Using this action allows the script to only indicate the elements that are changing.
- **fastUpdate**. This action will update the object but does not perform a subsequent 'read' to return the resulting view of the BO. This option is better than update for performance purposes if there is no reason to re-read the record.
- **delete**. This action deletes the object.
- **replace**. This action is an alternate to the update action. The replace action completely replaces existing BO data with the information in the request document. Typically, the replace action is used when a BO contains a list because it is easier to simply replace all instances of a list rather than attempt a list merge, which requires special logic to delete a list instance explicitly.

> **Note:** The replace action must be used when using the UI map functionality to Upload a CSV File.

> **Note:** The replace action is currently not supported for any maintenance object that is maintained in a 'fixed' page that uses a list metaphor to show all the records in the page at once. Currency is an example of this type of page.

Examples:

```
invokeBO -'BusinessObject' using -"dataArea" for fastAdd;
 -
invokeBO $variableBO using -"dataArea" for fastUpdate;
 -
invokeBO -"daName/boElement" using -"dataArea" for replace;
```

The syntax of the invoke statements for both a business service and service script are similar. The BS / SS is specified along with the XPath reference to the group name:

- **invokeBS 'BS Name' | $variable | "xpath" using "xpath";**
- **invokeSS 'SS Name' | $variable | "xpath" using "xpath";**

The examples use the **invokeBS** statement but the statements are similar for the **invokeSS** statement.

```
invokeBS -'BusinessService' using -"dataArea";
 -
invokeBS $variableBS using -"dataArea";
 -
invokeBS -"daName/bsElement" using -"dataArea";
```

BO Warnings. Note that for BPA scripting, the **invoke** statements may also indicate how to handle warnings.

Syntax	Description	Examples
with warn asError	Indicates that a warning should be treated as an error displayed in the UI map. The text **asError** is optional.	`invokeBO -'BusinessObject' - using -"dataArea" for add with warn - asError;`

Copyright © 2024, Oracle and/or its affiliates.

Syntax	Description	Examples
		`invokeSS -'ServiceScript' - using -"dataArea" with warn;`
with warn popup	Indicates that a warning should be presented in the standard framework popup. In this scenario, the user is presented with standard OK and Cancel buttons. If the user clicks OK, it means that the process should continue. If the user clicks Cancel, the processing should discontinue. This is the recommended setting.	`invokeBS -"daName/bsElement" - using -"dataArea" with warn popup;`

with warn suppress	Indicates that a warning should be suppressed. This is the default if no warning syntax is added to the **invoke** statement.	`invokeBS -"daName/bsElement" - using -"dataArea" with warn suppress;`
		
		`invokeSS -'ServiceScript' - using -"dataArea";`

Note: For service scripts, all objects invoked from the service script will inherit their warning level. Therefore, if the service script is invoked **with warn**, all nested invoke statements will also be invoked **with warn**.

For BPA scripts, there should also be logic following the invocation to handle errors and warnings (if **with warn as popup** is used). The system variable **$WARNING** is set to **true** if the user has clicked the Cancel button in the warning popup. If a map was shown, the logic should redisplay the map (where the warning message will be redisplayed.) This allows the user to potentially make changes and save again. If no map was shown prior to the warning, then the logic should terminate.

The system variable **$ERROR** indicates that an error was received. If a map was shown prior to the error, then the logic should redisplay the map where the error will be shown. If no map was shown, then the product provides a BPA Script **F1–HandleErr** that should be used to display the error. The following is an example of typical error handling logic.

```
invokeBO -"F1-DetermineBo/output/bo" using -"boSchema" for update with warn popup;
if ("$WARNING")
   if ("map_schema/action = -'DEL'")  -
     terminate;
   else
     goto maintMap;   -
   end-if;
end-if;
if ("$ERROR")
  if ("map_schema/action -!= -'DEL'")
    goto maintMap;
  else
    transferControl -'F1-HandleErr';
  end-if;
end-if;
```

System and Global Variables

The following tables highlight system and global variables available for script writing.

System Variables - All Script Types

Copyright © 2024, Oracle and/or its affiliates.

The following system variables are available for all script types (service scripts, plug-in scripts, and BPA scripts).

Variable	Description	Example

$BLANK	Represents an empty node.	`if ("string(parm/spouse/name) -!= $BLANK")`
		`  goto addSpouse;`
		`end-if;`

$CURRENT-DATE	Represents the current date. For BPA scripts, this is the browser date. For server scripts this is the server date (and is affected by the system date override logic).	`move -"$CURRENT-DATE" to -$tempDate;`
$CURRENT-STD-DTTM	Represents the current date-time expressed in standard time (meaning without any adjustments for summer time / daylight savings time).	`move -"$CURRENT-STD-DTTM" to -$tempDateTime;`
$DEVICE-OS	Represents the user's device operating system.	`move -"$DEVICE-OS" to -$tempDeviceOs;`
$DEVICE-BROWSER	Represents the user's device browser.	`move -"$DEVICE-BROWSER" to -$tempDeviceBrowser;`
$DEVICE-DISPLAY-TYPE	Represents the user's device screen display type whether it is Desktop size or Medium or Small size. Returned values may be like oraDesktop, oraTablet and oraPhone.	`move -"$DEVICE-DISPLAY-TYPE" to -$tempDeviceDisplayType;`
$DEVICE-INFO	Provides the combination of all three device properties (DEVICE-OS, DEVICE-BROWSER and DEVICE-DISPLAY-TYPE) and each property value is separated by semi-colon.	`move -"$DEVICE-INFO" to -$tempDeviceInfo;`

System Variables - BPA Scripts Only

Copyright © 2024, Oracle and/or its affiliates.

The following system variables are only available / applicable for BPA script types.

Variable	Description	Example
$DOUBLE_QUOTE	Represents a double quote.	`move -"$DOUBLE_QUOTE" to -$tempField;`
$SINGLE_QUOTE	Represents an apostrophe.	`move -"$SINGLE_QUOTE" to -$tempField;`
$SPACE	Contains a single space value.	`move -"$SPACE" to $tempField;`
$SYSTEM-DATE	Represents the server date. Note that this date is affected by the system date override logic)	`move -"$SYSTEM-DATE" to -$tempDate;`

System Variables - Server Scripts Only

The following system variables are only available / applicable for service script and plug-in script types.

Variable	Description	Example
$ADDITIONAL-IP-INFO	An HTTP request includes an "additional IP address" header field. This may be populated by an implementation if there is some information available on the proxy server or load balancer, such as the originating IP address.	`move -"$ADDITIONAL-IP-INFO" to -"parm/request/headerIpAddress";`
$CURRENT-DTTM	Represents the current date-time.	`move -"$CURRENT-DTTM" to -$tempDateTime;`
$F1-INSTALLATION-TIMEZONE	Represents the time zone code defined on the installation options.	`move -"$F1-INSTALLATION-TIMEZONE" to $timeZone;`
$LANGUAGE	Represents the language code the script is using. Typically this is the user's default language.	`move -"$LANGUAGE" to -$tempLanguage;`
$PROCESS-DATE	Represents the process date. The process date differs from the current date because the process date will remain consistent throughout the duration of the process being executed. For example, if a service script stores several business objects – the process date is initialized at the start of the service script execution and each business object will	`move -"$PROCESS-DATE" to -$tempDate;`

Copyright © 2024, Oracle and/or its affiliates.

Variable	Description	Example
	have the same process date defaulted. The current date, especially the current date time, will reflect the actual time of processing.	
$PROCESS-DTTM	Represents the process date-time. Note that the process date and time is initialized at the start of a particular process and will not reflect the exact date and time of an update.	`move -"$PROCESS-DTTM" to -$tempDateTime;`
$REQUESTING-IP-ADDRESS	Represents the IP address from the HTTP request. Note that if the request is routed through a proxy server or load balancer, this IP address is be the IP address of the proxy or load balancer, not the IP address of the end user. Refer to the **$ADDITIONAL-IP-INFO** variable for information.	`move -"$REQUESTING-IP-ADDRESS" to -"parm/request/systemIpAddress";`
$USER	Represents the user ID of the user executing the script.	`move -"$USER" to $tempUser;` For BPA scripts that need to know the user ID of the current user, refer to Useful Tips and Tricks.

Global Variables

BPA scripts and service scripts have access to the values defined in Global Context.

When a BPA script is launched from the user interface, these variables will be automatically initialized. They may be referenced with a single dollar sign in front of the field name. For example if PER_ID is a supported global variable, then $PER_ID can be referenced within the BPA script:

```
move -"$PER_ID" to -"schema/customerId";
```

For service scripts, global variables may only be referenced if the service script has been invoked directly from a BPA script or a zone on a portal. When a service script is invoked from a BPA script or portal zone, it will have access to the suite of global context variables populated in the UI session. For service scripting, the global fields must be prefixed by two dollar signs (instead of one like in BPA scripting). For example if PER_ID is a supported global context variable, then $$PER_ID can be referenced within the service script.

```
move $$PER_ID to -"schema/customerId";
```

Note: As described in Context Variables, a service script may declare context variables that use the same two dollar sign syntax.

Perform Script and Transfer Control Statements

The edit data step supports functionality analogous to the Perform script step type and the Transfer Control step type. These are both applicable only to BPA scripts.

Copyright © 2024, Oracle and/or its affiliates.

Syntax	Valid Values	Comments
performScript	'BPA Script Name'	Script to perform is explicitly provided.
	$Variable	Script to perform is found in a variable.
	"XPath"	Script to perform is found in an element, referenced by its XPath.
transferControl	Analogous to the **performScript** statement	

Note: When the script named in the **performScript** statement has finished, control will be returned to the calling BPA script. When the script named in the **transferControl** statement has finished, you will not be returned to the calling script, complete control will be granted to the transferred to script.

Navigate Statement

The edit data step supports functionality analogous to the Navigate to a page step type. This is applicable only to BPA scripts.

Syntax	Valid Values	Comments
navigate	'Navigation Code'	Navigation option is explicitly provided.
	$Variable	Navigation option is found in a variable.
	"XPath"	Navigation option is found in an element, referenced by its XPath.

In addition, the edit data step supports the ability to indicate that the sidebar should be refreshed when navigating. This is only applicable to BPA scripts.

Syntax	Valid Values
navigateAndReloadDashboard	Analogous to the **navigate** statement

Declare BO with BO Group

This statement is specific to BPA scripts that plan to use the base script Main BO Maintenance Processing (**F1–MainProc**) for its Generate Edit Map statements. This script expects that the data used to display in the map is within a **boGroup** tag.

Syntax	Valid Values	Comments
declareBOWithBOGroup	'BO Name'	BO is explicitly provided.
	$Variable	BO is found in a variable.
	"XPath"	BO is found in an element, referenced by its XPath.

The following table highlights additional syntax for this statement.

Syntax	Valid Values
as	'Dynamic Schema Name'

Copyright © 2024, Oracle and/or its affiliates.

Examples:

```
declareBOWithBOGroup -'BusinessObject' as -'newMapSchema';
 -
declareBOWithBOGroup $variableBO as -'newMapSchema';
 -
declareBOWithBOGroup -"daName/boElement" as -'newMapSchema';
```

Invoke Map Statement

The edit data step supports functionality analogous to the Invoke map step type. This is applicable only to BPA scripts.

Syntax	Valid Values	Comments
invokeMap	'Map Name'	UI Map is explicitly provided.
	$Variable	UI Map is found in a variable.
	"XPath"	UI Map is found in an element, referenced by its XPath.

The following table highlights additional syntax for this statement.

Syntax	Valid Values	Comments
using	"Data Area group name"	Indicates the data area to be passed to and from the server when rendering the HTML form associated with the map.
target	bpa	
	page	
	popup	

Refer to the Invoke map step type for more information about the **target** values.

If the UI map is configured to return a value, then it can be evaluated using the **$MAP-VALUE** variable.

```
invokeMap -'UI Map' using -"dataArea";
 -
invokeMap $variableMap using -"dataArea";
 -
invokeMap -"daName/mapElement" using -"dataArea" target bpa;
 -
// $MAP-VALUE is a variable returned by the invoked map.
if ("$MAP-VALUE='continue' -")
    goto 300;
else
    terminate;
end if;
```

Generate Edit Map Statements

The 'generate edit map' statements are used to dynamically generate and launch a UI edit map based on a schema definition. The schema used may be a BO schema, a BS schema, an SS schema or a DA schema. This is applicable only to BPA scripts. The generate statement will differ based on the type of schema, but the syntax is analogous.

Copyright © 2024, Oracle and/or its affiliates.

Syntax
generateBOEditMap
generateBSEditMap
generateSSEditMap
generateDAEditMap

The BO code / BS code / SS code / DA code may be specified using a literal (surrounded by single quotes), as a temporary variable or by referencing an XPath schema location (surrounded by double quotes).

The following table highlights additional syntax for this statement.

Syntax	**Valid Values**	**Comments**
using	"Data Area group name"	Indicates the data area to be passed to and from the server when rendering the HTML form associated with the map.
target	bpa	
	page	
	popup	

The target values indicate where the generated map should be displayed as described in the Invoke map step type. If the UI map is configured to return a value, then it can be evaluated using the **$MAP-VALUE** variable.

The examples use the **generateBOEditMap** but the statements are similar for the other schema types.

```
generateBOEditMap -'BO Name' using -"dataArea";
 -
generateBOEditMap $variableMap using -"dataArea";
 -
generateBOEditMap -"daName/mapElement" using -"dataArea" target bpa;
 -
// $MAP-VALUE is a variable returned by the invoked map.
if ("$MAP-VALUE='continue' -")
    goto 300;
else
    terminate;
end if;
```

Terminate Statement

The edit data step supports functionality analogous to the Terminate step type.

The following is an example of a simple **terminate** step that will stop the script.

```
if ("not(parm/spouse/name)")
  terminate;
else
  goto addSpouse;
end-if;
```

The **terminate with error** statement is only available in a service script.

Note: For issuing an error from a BPA script refer to Useful Tips and Tricks.

Copyright © 2024, Oracle and/or its affiliates.

Syntax	Attributes	Comments
terminate with error (x, y %n= element=)	'x' represents the message category	Required.
	'y' represents the message number	Required.
	%n="Element XPath" or %n='literal'	Specify the substitution parameters supported by the message using either literal values or XPath references. When the value to substitute is an element within a list, use the XPath of the element within the list qualified by the list occurrence.
	element='Element XPath' or element=$variable	Optionally specify an element name within a UI map to highlight as part of the error. When the element in error is within a list, use the XPath of the element within the list qualified by the list occurrence.

Example of a simple field:

```
if ("string(customer/lastName) = $BLANK")
  terminate with error (8000, 1001 %1="customer/lastName" %2='Last name required' -
element='customer/lastName');
end-if;
```

Example of terminate where the element to mark is in a list:

```
for ($list in -"parm/hard/newBusinessObject/listName") -
  if -//** check some condition for elementName
      terminate with error (11000, 11000 %1="$list/elementName" element='$list/
elementName');
  end-if;
end-for;
```

Fastpath: For more information on presenting errors in a UI map, please refer to Display Errors.

Invoking Groovy Code

If you have a plug-in script or a service script that mixes XPath scripting and Groovy scripting, the edit data step supports the ability to invoke Groovy code using the syntax **invokeGroovy 'method';** where 'method' is the name of a method defined in a **Groovy Members** step within the script. Only methods that do not receive arguments and return void can be invoked in this way. However, the method invoked from the edit data step can be supported by additional Groovy code in other **Groovy Members** step types.

Example Edit Data step:

```
invokeGroovy -'invoke';
```

Example Groovy Members step:

```
void invoke() {
  initParms()
  readBO()
  initConfig()
  retrieve()
  updateBO()
}
```

Note: As mentioned in Using Groovy Within Scripts, if this is a plug-in script and the code is only Groovy, you should use the **Groovy** script engine version rather than the above technique. This is considered more efficient as it avoids the conversion of the data to and from the XML interface.

Copyright © 2024, Oracle and/or its affiliates.

Debugging a BPA Script

If a BPA script has height greater than zero, then selected nodes of the script's data area can be displayed at runtime. The XML data is displayed during script execution within the BPA script's display area. Specify the XPath of an XML node from any of the BPA script's data areas, between the paired characters: '%+' and '+%'.

For example, the entire contents of the schema group node named 'input', and the specific contents of the schema element named 'output/status' will be displayed in the BPA script's display area. The debug text must be entered into the BPA script's text area and not within the script's edit data field. Debug text can be declared for any explicit step of the script.

```
display input: %+input+% -, and output status: %+output/status+%
```

Useful Tips and Tricks

This section provides some suggestions about how to implement some common logic.

Finding the Logged In User in a BPA

For server based scripts, the **$USER** variable is populated with the current logged in user. This variable is not available for BPA scripts. The BPA can call the service script **F1-GetUser** if this information is needed.

Issuing an Error from a BPA script

For server based scripts, you use the **terminate with error** statement to return an error message to display to the user. The BPA can display that error by transferring control to **F1–HandleErr** as described in the Declare and Invoke Schema Based Objects. This is not supported in a BPA script. A technique that is available to a BPA for issuing an error is to call the business service **F1-RethrowError** passing the error details. This business service populates the appropriate system error fields needed to issue the error. Then transfer control to **F1–HandleErr**.

```
  if ("string($assignedToUser) -!= string(F1-GetUser/user)")
    declareBS -'F1-RethrowError' as -'errorMsg';
    move -'11010' to -"errorMsg/messageCategory";
    move -'11511' to -"errorMsg/messageNumber";
    move -"$toDoEntryId" to -"errorMsg/messageParameters/+parameters/
parameterValue";
    invokeBS -'F1-RethrowError' using -"errorMsg";
    transferControl -'F1-HandleErr';  -
  end-if;
```

Go To Steps

Go to steps allow you to jump to a step other than the next step. Additional fields required for **Go To** steps:

Next Step defines the step to which the script should jump.

Height Steps

Note: BPA scripts only. This step type is only applicable to BPA scripts.

Height steps are used to change the height of the script area to be larger or smaller than the standard size.

The following additional fields are required for **Height** steps:

Script Window Height defines the number of **Pixels** or the **Percentage** (according to the **Height Unit**) that the script window height should be adjusted. The percentage indicates the percentage of the visible screen area that the script area uses. For example, a percentage value of **100** means that the script area will use the entire area.

Note: Standard Number of Pixels. The default number of pixels used by the script area is **75**.

Note: Adjust script height in the first step. If you want to adjust the height of the script area, it is recommendation to define the **height** step type as your first step. Otherwise, the script area will open using the standard height and then readjust, causing the screen to redisplay.

Copyright © 2024, Oracle and/or its affiliates.

Note: Hide script area. You could use this type of step to set the height to **0** to hide the script area altogether. This is useful if the script does not require any prompting to the user. For example, perhaps you define a script to take a user to a page and with certain data pre-populated and that is all.

Note: Automatically close script area. If you want the script area to close when a script is completed, you could define the final step type with a height of 0.

Input Data Steps

Note: BPA scripts only. This step type is only applicable to BPA scripts.

Input data steps cause the user to be prompted to populate an input field in the script area. The input value can be saved in a field on a page or in temporary storage. A **Continue** button always appears adjacent to the input field. You may configure steps of this type to display one or more buttons in addition to the **Continue** button. For example, you may want to provide the ability for the user to return to a previous step to fix incorrect information. The user may click on any of these buttons when ready for the script to continue.

The following additional fields are required for **Input Data** steps:

Destination Field Type and **Destination Field Name** define where the input field will be saved. The **Field Type** defines where the field is located. The **Field Name** defines the name of the field. The following points describe each field type:

- **Page Data Model**. Use this field type to put the input field into a field that resides on any of the tab pages in the object display area (i.e., the field populated doesn't have to reside on the currently displayed tab page, it just has to be part of the object that's currently displayed). Refer to How To Find The Name Of Page Data Model Fields for instructions on how to find the appropriate **Field Name**.
- **Temporary Storage**. Use this field type to put the input field into temporary storage. Use **Field Name** to name the field in temporary storage. Refer to How To Name Temporary Storage Fields for more information.
- **User Interface Field**. Use this field type to put the input field into a field that resides on the currently displayed tab page. Note, if you want to execute underlying default logic, you must populate a **User Interface Field**. Refer to How To Find The Name Of User Interface Fields for instructions on how to find the appropriate **Field Name**.

The **Prompt Values** grid may be used to define additional buttons. A separate button is displayed in the script area for each entry in this grid.

- **Prompt Text** is the verbiage to appear on the button. Refer to How To Substitute Variables In Text for a description of how you can substitute field values into the prompts.
- **Sequence** controls the order of the buttons.
- **Next Script Step** defines the step to execute if the user clicks the button.

Invoke Business Object Steps

Invoke business object steps allow you to interact with a business object in order to obtain or maintain its information.

The following additional fields are required for **Invoke business object** steps:

Use **Warning Level** to indicate whether warnings should be suppressed and if not, how they should be presented to the user. By default, warnings are suppressed. If **Warn As Popup** is used, the warning is displayed using the standard popup dialog. If **Warn As Error** is used processing is directed to the **If Error, Go To** step. This field is only applicable to BPA scripts. Refer to BO warnings for more information about handling warnings when invoking a BO.

Group Name references the data area to be passed to and from the server when communicating with the **Business Object**. Indicate the **Action** to be performed on the object when invoked. Valid values are **Add, Delete, Fast Add (No Read), Fast Update (No Read), Read, Replace, Update**.

Fastpath: Refer to BO Actions for more information about the various actions.

The business object call will either be successful or return an error. The next two fields only appear when the call is issued from a BPA script, to determine the step to which control is passed given the outcome of the call.

If Success, Go To defines the step that is executed if the call is successful. This field is only applicable to BPA scripts.

Copyright © 2024, Oracle and/or its affiliates.

If Error, Go To defines the step that is executed if the call returns on error. Please note that the error information is held in global variables. This field is only applicable to BPA scripts.

> **Note: Error technique.** Let's assume a scenario where a business object is invoked from a BPA script and the call returned an error. If the BPA script is configured to communicate with the user using a UI map, you may find it useful to invoke the map again to present the error to the user. Alternatively, you may invoke a step that transfers control to a script that displays the error message information and stops.

Invoke Business Service Steps

Invoke business service steps allow you to interact with a business service.

The following additional fields are required for **Invoke business service** steps:

Use **Warning Level** to indicate whether warnings should be suppressed and if not, how they should be presented to the user. By default, warnings are suppressed. If **Warn As Popup** is used, the warning is displayed using the standard popup dialog. If **Warn As Error** is used processing is directed to the **If Error, Go To** step. This field is only applicable to BPA scripts.

Group Name references the data area to be passed to and from the server when the **Business Service** is invoked.

The business service call will either be successful or return an error. The next two fields only appear when the call is issued from a BPA script, to determine the step to which control is passed given the outcome of the call.

If Success, Go To defines the step that is executed if the call is successful. This field is only applicable to BPA scripts.

If Error, Go To defines the step that is executed if the call returns on error. Please note that the error information is held in global variables. This field is only applicable to BPA scripts.

> **Note: Error technique.** Let's assume a scenario where a business service is invoked from a BPA script and the call returned an error. If the BPA script is configured to communicate with the user using a UI map, you may find it useful to invoke the map again to present the error to the user. Alternatively, you may invoke a step that transfers control to a script that displays the error message information and stops.

Invoke Function Steps

> **Note:** Functions were implemented prior to the introduction of business services (BS), service scripts (SS) and business objects (BO). The functionality is still supported, but the recommendation for implementations going forward is to use a step that invokes one of the above configuration tool objects in a script rather than defining a function.

> **Note: BPA scripts only.** This step type is only applicable to BPA scripts.

Invoke function steps may be used to retrieve or update data independent of the page currently being displayed. For example, if you design a script that takes different paths based on the customer's customer class, you could invoke a function to retrieve the customer's customer class.

> **Fastpath:** You must set up a function before it can be referenced in a script. Refer to Maintaining Functions for the details.

The following additional fields are required for **Invoke Function** steps:

Function defines the name of the function. The function's **Long Description** is displayed below.

When a function is invoked, it will either be successful or return an error. The next two fields control the step to which control is passed given the outcome of the function call:

- **If Success, Go to** defines the step that is executed if the function is successful.
- **If Error, Go to** defines the step that is executed if the function returns on error. Refer to How To Use Constants In Scripts for a list of the global variables that are populated when a function returns an error.

Copyright © 2024, Oracle and/or its affiliates.

> **Note: Error technique.** If a function returns an error, we recommend that you invoke a step that transfers control to a script that displays the error message information and stops (note, the error information is held in global variables). You would invoke this script via a **Transfer Control**.

The **Send Fields** grid defines the fields whose values are sent to the function and whose field value source is not **Defined On The Function**. For example, if the function receives an account ID, you must define the name of the field in the script that holds the account ID.

- **Field** contains a brief description of the field sent to the function.
- **Source Field Type** and **Mapped Field / Value** define the field sent to the function. Refer to the description of Source Field under How To Set Up Move Data Steps for a description of each field type.
- **Comments** contain information about the field (this is defined on the function).

The **Receive Fields** grid defines the fields that hold the values returned from the function. For example, if the function returns an account's customer class and credit rating, you must set up two fields in this grid.

- **Field** contains a brief description of the field returned from the function.
- **Destination Field Type** and **Mapped Field** define the field returned from the function. Refer to the description of Destination Field under How To Set Up Move Data Steps for a description of each field type.
- **Comments** contain information about how the field (this is defined on the function).

Invoke Map Steps

> **Note: BPA scripts only.** This step type is only applicable to BPA scripts.

Invoke map steps are used to invoke a UI Map to display, capture and update data using an HTML form. You may configure steps of this type to display one or more buttons in addition to the **Continue** button. For example, you may want to provide the ability for the user to return to a previous step to fix incorrect information. The user may click on any of these buttons when ready for the script to continue.

The following additional fields are required for **Invoke map** steps:

Group Name references the data area to be passed to and from the server when rendering the HTML form associated with the **Map**.

Use **Target Area** to designate where the map will be presented.

- Select **BPA Zone** if the map should be presented within the script area.
- Select **Page Area** if the map should be presented in the object display area, i.e. the frame typically used to house a maintenance page.
- Select **Pop-up Window** if the map should be launched in a separate window.

The **Returned Values** grid contains a row for every button defined on the map.

- **Returned Value** is the value returned when the user clicks the button.
- **Use as Default** can only be turned on for one entry in the grid. If this is turned on, this value's Next Script Step will be executed if the returned value does not match any other entry in the grid. For example, if the user closes a pop-up (rather than clicking a button), the default value will be used.
- **Next Script Step** defines the step to execute if the user clicks the button.

Invoke Service Script Steps

Invoke service script steps allow you to execute a service script.

The following additional fields are required for **Invoke service script** steps:

Use **Warning Level** to indicate whether warnings should be suppressed and if not, how they should be presented to the user. By default, warnings are suppressed. If **Warn As Popup** is used, the warning is displayed using the standard popup dialog. If **Warn As Error** is used processing is directed to the **If Error, Go To** step. This field is only applicable to BPA scripts.

Group Name references the data area to be passed to and from the server when the **Service Script** is invoked.

Copyright © 2024, Oracle and/or its affiliates.

The service script call will either be successful or return an error. The next two fields only appear when the call is issued from a BPA script to determine the step to which control is passed given the outcome of the call.

If Success, Go To defines the step that is executed if the call is successful. This field is only applicable to BPA scripts.

If Error, Go To defines the step that is executed if the call returns on error. Please note that the error information is held in global variables. This field is only applicable to BPA scripts.

Note: Error technique. Let's assume a scenario where a service script is invoked from a BPA script and the call returned an error. If the BPA script is configured to communicate with the user using a UI map, you may find it useful to invoke the map again to present the error to the user. Alternatively, you may invoke a step that transfers control to a script that displays the error message information and stops.

Groovy Imports Steps

Note: Server based scripts only. This step type is not applicable to BPA scripts.

Groovy Imports step types provide a free format text area where you list the Groovy classes to be imported for use by the code in **Groovy Member** steps within the script.

Enter a description of the imports step in the **Text** field. Click the adjacent icon to open a window providing more space for entering text.

Enter the list of classes to import in the **Edit Data Text** field using the syntax **import 'class';** where 'class' is the fully qualified package name of the Groovy class. Click the adjacent icon to open a window providing more space for editing the text.

When saving a script, if the system has been configured accordingly, when the system detects references to a Groovy class that is not explicitly listed in this step, the system warns the user and adds an explicit import for that class. (If no Groovy Import step type exists for the script, one will be added.) Users should review the imported class to ensure it is the expected class, in the case of multiple classes with a similar name.

Note: For security, the classes that may be imported are restricted to those allowed by the Framework. The warning and automatic update to the script is controlled by a property setting. Refer to the topic Using Groovy Within Scripts for more information.

Groovy Library Interface Steps

Note: Server based scripts only. This step type is not applicable to BPA scripts.

Groovy Library Interface steps are only applicable to **Groovy Library Scripts**. They provide a free format text area where you can list the Groovy methods defined within the script that are available for use by other scripts.

Enter a description of the interface step in the **Text** field. Click the adjacent icon to open a window providing more space for entering text.

Enter the list of interface methods in the **Edit Data Text** field. Click the adjacent icon to open a window providing more space for editing the list.

Note: Every **Groovy Library Script** must have one and only one step of type **Groovy Library Interface**. The supporting code for the available methods is defined using one or more **Groovy Member** steps in the same script. Refer to the topic Using Groovy Within Scripts for more information.

Groovy Member Steps

Note: Server based scripts only. This step type is not applicable to BPA scripts.

Groovy Member steps provide a free format text area where you can enter Groovy code.

Enter a description of the code block in the **Text** field. Click the adjacent icon to open a window providing more space for entering text.

Copyright © 2024, Oracle and/or its affiliates.

Enter your code in the **Edit Data Text** field. Click the adjacent icon to open a window providing more space for editing the code.

> **Note:** While it is possible to set up multiple steps of type **Groovy Member** the system treats these steps as a single class for compilation and execution purposes. Refer to the topic Using Groovy Within Scripts for more information.

Label Steps

Label steps allow you to describe what the next step(s) are doing. Steps of this type are helpful to the script administrators when reviewing or modifying the steps in a script, especially when a script has many steps. When designing a script, the label steps enable you to provide a heading for common steps that belong together. The script tree displays steps of this type in a different color (green) so that they stand out from other steps.

There are no additional fields for **Label** steps.

Mathematical Operation Steps

> **Note: BPA scripts only.** This step type is only applicable to BPA scripts.

Mathematical operation steps allow you to perform arithmetic on fields. You can also use this type of step to add and subtract days from dates. For example, you could calculate a date 7 days in the future and then use this value as the customer's next credit review date. The following additional fields are required for **Mathematical Operation** steps:

Base Field Type and **Base Field Name** define the field on which the mathematical operation will be performed. The **Field Type** defines where the field is located. The **Field Name** defines the name of the field. The following points describe each field type:

- **Page Data Model**. Use this field type when the field resides on any of the tab pages in the object display area. Refer to How To Find The Name Of Page Data Model Fields for instructions on how to find the appropriate **Field Name**.
- **Temporary Storage**. Use this field type when the field resides in temporary storage. You must initialize the temporary storage field with a Move Data step before performing mathematical operations on the field. Refer to How To Set Up Move Data Steps for more information.
- **User Interface Field**. Use this field type when the field resides on the currently displayed tab page. Refer to How To Find The Name Of User Interface Fields for instructions on how to find the appropriate **Field Name**.

Math Operation controls the math function to be applied to the **Base Field**. You can specify +, -, /, and *. Note, if the base field is a date, you can only use + or -.

Math Field Type, **Math Field Name** and **Math Field Value** define the field that contains the value to be added, subtracted, divided, or multiplied. The following points describe each field type:

- **Current To Do Information**. Use this field type when the value resides on the current To Do entry. Refer to How To Use To Do Fields for instructions on how to define the appropriate **Field Name**.
- **Page Data Model**. Use this field type when the value resides on any of the tab pages in the object display area. Refer to How To Find The Name Of Page Data Model Fields for instructions on how to find the appropriate **Field Name**.
- **Predefined Value**. Use this field type when the value is a constant. When this field type is used, use **Source Field Value** to define the constant value. Refer to How To Use Constants In Scripts for more information. Note, if you are performing arithmetic on a date, the field value must contain the number and type of **days**/ **months**/ **years**. For example, if you want to add 2 years to a date, the source field value would be **2 years**.
- **Temporary Storage**. Use this field type when the value is a field that you put into temporary storage in an earlier step. The **Field Name** must be the same as defined in an earlier step.
- **User Interface Field**. Use this field type when the value resides in a field on the current tab page. Refer to How To Find The Name Of User Interface Fields for instructions on how to find the appropriate **Field Name**.

Move Data Steps

Move data steps allow you to move data (from a source to a destination). The following additional fields are required for **Move data** steps:

Source Field Type, **Source Field Name** and **Source Field Value** define what you're moving. The following points describe each field type:

Copyright © 2024, Oracle and/or its affiliates.

- **Context Variable**. Use this field type in a plug-in or service script if the source value is a variable initiated in a higher level script. This is only applicable to Service Scripts and Plug-in Scripts.
- **Current To Do Information**. Use this field type when the source value resides on the current To Do entry. Refer to How To Use To Do Fields for instructions on how to define the appropriate **Field Name**. This is only applicable to BPA Scripts.
- **Data Area**. Use this field type when the field being compared is one that you put into one of the script's data areas in an earlier step. **Field Name** must reference both a data area structure name as well as the field, for example "parm/charType". Refer to How To Reference Fields In Data Areas for instructions on how to construct the appropriate **Field Name**.
- **Global Context**. Use this field type when the source value is a global context variable. This is only applicable to BPA Scripts.
- **Page Data Model**. Use this field type when the source value resides on any of the tab pages in the object display area (i.e., the source field doesn't have to reside on the currently displayed tab page, it just has to be part of the object that's currently displayed). Refer to How To Find The Name Of Page Data Model Fields for instructions on how to find the appropriate **Field Name**. This is only applicable to BPA Scripts.
- **Portal Context**. Use this field type when the source value is a variable in the portal context. This is only applicable to BPA Scripts.
- **Predefined Value**. Use this field type when the source value is a constant value defined in the script. When this field type is used, use **Source Field Value** to define the constant value. Refer to How To Use Constants In Scripts for instructions on how to use constants.

Note: Concatenating fields together. You can also use **Predefined Value** if you want to concatenate two fields together. For example, let's say you have a script that merges two persons into a single person. You might want this script to change the name of the person being merged out of existence to include the ID of the person remaining. In this example, you could enter a **Source Field Value** of **%ONAME merged into person %PERID** (where **ONAME** is a field in temporary storage that contains the name of the person being merged out of existence and **PERID** contains the ID of the person being kept). Refer to How To Substitute Variables In Text for a description of how you can substitute field values to compose the field value.

- **Temporary Storage**. Use this field type when the source value is a field that you put into temporary storage in an earlier step. The **Field Name** must be the same as defined in an earlier step.
- **User Interface Field**. Use this field type when the source value resides on the currently displayed tab page. Refer to How To Find The Name Of User Interface Fields for instructions on how to find the appropriate **Field Name**. This is only applicable to BPA Scripts.

Destination Field Type and **Destination Field Name** define where the source field will be moved. The **Field Type** defines where the field is located. The **Field Name** defines the name of the field. The following points describe each field type:

- **Context Variable**. Use this field type in your plug-in or service script if you use a variable to communicate information to a lower level service script or schema. This is not applicable to BPA Scripts.
- **Data Area**. Use this field type when the destination field resides on one of the scripts data areas. **Field Name** must reference both a data area structure name as well as the field, for example "parm/charType". Refer to How To Reference Fields In Data Areas for instructions on how to construct the appropriate **Field Name**.
- **Page Data Model**. Use this field type when the destination field resides on any of the tab pages in the object display area (i.e., the field populated doesn't have to reside on the currently displayed tab page, it just has to be part of the object that's currently displayed). Refer to How To Find The Name Of Page Data Model Fields for instructions on how to find the appropriate **Field Name**. This is only applicable to BPA Scripts.
- **Portal Context**. Use this field type when the destination to be updated is in the current portal context. This is only applicable to BPA Scripts.
- **Temporary Storage**. Use this field type when the destination field resides in temporary storage. Use **Field Name** to name the field in temporary storage. Use **Field Name** to name the field in temporary storage. Refer to How To Name Temporary Storage Fields for more information.
- **User Interface Field**. Use this field type when the destination field resides on the currently displayed tab page. Refer to How To Find The Name Of User Interface Fields for instructions on how to find the appropriate **Field Name**. This is only applicable to BPA Scripts.

Copyright © 2024, Oracle and/or its affiliates.

Note: Conditional field types. The field types **Current To Do Information**, **Page Data Model** and **User Interface Field** are only applicable to BPA scripts.

Navigate To A Page Steps

Note: BPA scripts only. This step type is only applicable to BPA scripts.

Navigate to a page steps cause a new page (or tab within the existing page) to be displayed in the object display area. Steps of this type are a precursor to doing anything on the page. The following additional field is required for **Navigate to a page** steps:

Navigation Option defines the transaction, tab, access mode (add or change) and any context fields that are passed to the transaction in change mode. For example, if you want a script to navigate to Person - Characteristics for the current person being displayed in the sidebar, you must set up an appropriate navigation option. Refer to Defining Navigation Options for more information.

Note: Navigating to a page in update mode. Before you can navigate to a page in change mode, the page data model must contain the values to use for the navigation option's context fields. If necessary, you can move values into the page data model using a Move Data step first. For example, before you can navigate to a page in change mode with an account ID in context, you may need to move the desired account ID into the ACCT_ID field in the page data model. The actual field name(s) to use are listed as context fields on the navigation option.

Perform Script Steps

Note: BPA scripts only. This step type is only applicable to BPA scripts.

Perform script steps cause another BPA script to be performed. After the performed script completes, control is returned to the next step in the original script. You might want to think of the scripts referred to on steps of this type as "subroutines". This functionality allows you to encapsulate common logic in reusable BPA scripts that can be called from other BPA scripts. This simplifies maintenance over the long term.

The following additional field is required for **Perform script** steps:

Subscript is the name of the script that is performed.

Press A Button Steps

Note: BPA scripts only. This step type is only applicable to BPA scripts.

Press a button steps cause a button or link text to be ‘pressed’ in the object display area, the application toolbar or the page title area. For example, you could use this type of step to add a new row to a person's characteristic (and then you could use a **Move Data** step to populate the newly added row with a given char type and value). The following additional fields are required for **Press a button** steps:

Button Name is the name of the button to be pressed. This button must reside on the currently displayed tab page (or in the application toolbar or page actions toolbar). Refer to How To Find The Name Of A Button for more information.

Prompt User Steps

Note: BPA scripts only. This step type is only applicable to BPA scripts.

Prompt user steps cause the user to be presented with a menu of options. The options can be presented using either buttons or in the contents of a drop down. You can also use steps of this type to pause a script while the user checks something out (and when the user is ready to continue with the script, they are instructed to click a prompt button). The following additional fields are required for **Prompt User** steps:

Prompt Type controls if the prompt shown in the script area is in the form of **Button(s)** or a **Dropdown**. Note, if you use a **Dropdown**, a Continue button appears adjacent to the dropdown in the script area when the step executes. The user clicks the Continue button when they are ready for the script to continue.

The **Prompt Values** grid contains a row for every value that can be selected by a user. Note, if you use a **Prompt Type** of **Button(s)**, a separate button is displayed in the script area for each entry in this grid.

Copyright © 2024, Oracle and/or its affiliates.

- **Prompt Text** is the verbiage to appear on the button or in the dropdown entry. Refer to How To Substitute Variables In Text for a description of how you can substitute field values into the prompts.
- **Sequence** controls the order of the buttons or dropdown entries.
- **Use As Default** can only be turned on for one entry in the grid. If this is turned on for a dropdown entry, this value is defaulted in the grid. If this is turned on for a button, this button becomes the default (and the user should just have to press `Enter` (or `space`) rather than click on it).
- **Next Script Step** defines the step to execute if the user clicks the button or selects the dropdown value.

Set Focus To A Field Steps

Note: BPA scripts only. This step type is only applicable to BPA scripts.

Set focus to a field steps cause the cursor to be placed in a specific field on a page. A **Continue** button always appears in the script area when this type of step executes. The user may click the **Continue** button when they are ready for the script to continue. You may configure steps of this type to display one or more buttons in addition to the **Continue** button. For example, you may want to provide the ability for the user to return to a previous step to fix incorrect information. The user may click on any of these buttons when ready for the script to continue.

The following additional fields are required for **Set focus to a field** steps:

Destination Field Name defines the field on which focus should be placed. This field must reside on the currently displayed tab page. Refer to How To Find The Name Of User Interface Fields for instructions on how to find the appropriate **Field Name**.

The **Prompt Values** grid may be used to define additional buttons. A separate button is displayed in the script area for each entry in this grid.

- **Prompt Text** is the verbiage to appear on the button. Refer to How To Substitute Variables In Text for a description of how you can substitute field values into the prompts.
- **Sequence** controls the order of the buttons.
- **Next Script Step** defines the step to execute if the user clicks the button.

Terminate Steps

Terminate steps cause a server-based script to end processing successfully or issue an error.

The following additional fields are required for **Terminate** steps:

Error indicates whether an error should be thrown or not. If error, **Error Data Text** must be specified, indicating the error message and any message substitution parameters. Refer to Edit Data Syntax the actual syntax of initiating an error message.

Note: The ability to terminate a step in error is only supported for server-based scripts.

Transfer Control Steps

Note: BPA scripts only. This step type is only applicable to BPA scripts.

Transfer control steps cause the current BPA script to terminate and the control to pass to another BPA script. You might want to construct a BPA script with steps of this type when the script has several potential logic paths and you want to segregate each logic path into a separate BPA script (for ease of maintenance).

The following additional fields are required for **Transfer control** steps:

Subscript is the name of the script to which control is transferred.

Additional Topics

The contents of this section provide additional information about steps.

How To Find The Name Of User Interface Fields

This section is only relevant if you are using the legacy capability of BPA scripts to bring a user to a base fixed page and populate information into fields on that page.

Copyright © 2024, Oracle and/or its affiliates.

> **Note:** For portal pages, this technique is not necessary. Script writers have the flexibility to populate information directly into the schema element for the business object and invoke UI maps to prompt for information.

Follow these steps to find the name of a field that resides on a fixed page:

- Navigate to the page in question.
- Right click in the body of the page.
- Select **Inspect** from the pop-up menu to display the HTML.
- In the search look for the text "Widget Info". Note that if there are multiple sections on the page, you will find multiple "Widget Info" results. Based on the details shown, you are able to confirm which section you are looking for. For example, the following is an example from the Algorithm - Main page:

```
* Widget Info:
*     Widget_ID , Element Type - label info - label
*     ALG_CD, IT - CI_ALG$ALG_CD -  Algorithm Code
*     IM_ALG_CD, IM - $SEARCH_FOR_ALG_LBL -  Search for Algorithm
*     OWNER_FLG, IS - $OWNER_FLG -  Owner
*     DESCR50, IT - $DESCR50 -  Description
*     IM_Main_algtypCont, IM - $F1_SHOW_ALG_CTX -  Show Algorithm Context Menu
*     ALG_TYPE_CD, IT - $ALG_TYPE_CD -  Algorithm Type
*     IM_ALG_TYPE_CD, IM - $SEARCH_ALGTYPE_LBL -  Search for Algorithm Type
*     ALG_TYPE_DESCR50, PL - $DESCR50 -  Description
*     ALG_TYPE_LONG_DESC, IA - $DESCRLONG -  Detailed Description
*     IM_VerScroll_lftArrow, IM - $PREVIOUS_ALG_LBL -  Previous Version
*     ALG_VER$recordCount, SN - $OF_LBL -  Of
*     IM_VerScroll_rtArrow, IM - $NEXT_ALG_LBL -  Next Version
*     IM_ADD, IM - $ADD_ALG_LBL -  Add Version
*     IM_DEL, IM - $DELETE_ALG_LBL -  Delete  Version
*     ALG_VER$EFFDT, IT - $EFFDT -  Effective Date
*     L_OWNER_LBL, LP - $OWNER_FLG -  Owner
*     ALG_VER$OWNER_FLG, IS - $OWNER_FLG -  Owner
*     resultGrid, GD
```

The field names that you'll reference in your scripts are defined on the left side of the HTML (e.g., ALG_CD, DESCR50, ALG_TYPE_CD, etc.).

The names of fields that reside in scrolls are in a slightly different format. The following is an example of the HTML for the version scroll that appears on Algorithm - Main. Notice that the fields in the scroll are prefixed with the name of the scroll plus a **$** sign. For example, the effective date is called **ALG_VER$EFFDT**.

```
*     IM_VerScroll_lftArrow, IM - $PREVIOUS_ALG_LBL -  Previous Version
*     ALG_VER$recordCount, SN - $OF_LBL -  Of
*     IM_VerScroll_rtArrow, IM - $NEXT_ALG_LBL -  Next Version
*     IM_ADD, IM - $ADD_ALG_LBL -  Add Version
*     IM_DEL, IM - $DELETE_ALG_LBL -  Delete  Version
*     ALG_VER$EFFDT, IT - $EFFDT -  Effective Date
*     L_OWNER_LBL, LP - $OWNER_FLG -  Owner
*     ALG_VER$OWNER_FLG, IS - $OWNER_FLG -  Owner
```

The names of fields that reside in grids are in a slightly different format. The following is an example of the HTML for the algorithm parameters grid that appears on Algorithm - Main. Notice that the fields in the grid are prefixed with the name of the grid plus a **:x$**. For example, the algorithm parameter label is called **ALG_PARM:x$ALG_PARM_LBL**. When you reference such a field in your script, you have the following choices:

- Substitute **x** with the row in the grid (and keep in mind, the first row in a grid is row **0** (zero); this means the second row is row **1**).
- If you want to reference the "current row" (that is, the row in which the cursor will be placed), you can keep the **x** notation (**x** means the "current row").

Copyright © 2024, Oracle and/or its affiliates.

```
* Widget Info:
*     Widget_ID , Element Type - label info - label
*     ALG_PARM:x$ALG_PARM_LBL, IL - CI_ALG_TYPE_PRM_L$ALG_PARM_LBL -  Parameter
*     ALG_PARM:x$SEQNO, IL - $SEQNO -  Sequence
*     ALG_PARM:x$ALG_PARM_VAL, IA - $ALG_PARM_VAL -  Value
*
```

How To Find The Name Of Page Data Model Fields

You find the name of a **Page Data Model** field in the same way described under How To Find The Name Of User Interface Fields. The only restriction is that you cannot refer to hidden / derived fields. However, you can refer to any of the object's fields regardless of the tab page on which they appear. For example, if you position the object display area to the Main tab of the Account transaction, you can reference fields that reside on all of the tab pages.

CAUTION: If you populate a **Page Data Model** field, none of the underlying default logic takes place. For example, if you populate a customer contact's contact type, none of the characteristics associated with the customer contact type are defaulted onto the customer contact. If you want the underlying defaulting to take place, you must populate a **User Interface Field**.

How To Find The Name Of A Button

This section is only relevant if you are using the legacy capability of BPA scripts to bring a user to a base fixed page and perform functionality by executing existing buttons on that page.

If you want a **Press a button** step to press a button or click a link in the application toolbar, use one of the following names:

Button Name
IM_GOBACK
IM_HISTORY
IM_GOFORWARD
IM_menuButton
IM_USER_HOME
IM_MY_PREF
IM_helpButton

If you want a **Press a button** step to press a button in the page actions toolbar, use one of the following names:

Button Name
IM_SAVE
IM_REFRESH
IM_CLEAR
IM_COPY
IM_DELETE
IM_ScrollBack
IM_ScrollForward

The following buttons are also supported:

Copyright © 2024, Oracle and/or its affiliates.

Button Name	Comments
IM_TO_DO	
IM_CurrentTo Do	This navigates to the To Do Entry page for the user's current To Do. Refer to Working On A To Do for more information.
IM_MINIMIZE_DASHBOARD	Pressing this will collapse the sidebar. Note that when a script is finished, it will return the sidebar to the state it was when the script was launched.
IM_MAXIMIZE_DASHBOARD	Pressing this will expand the sidebar. Note that when a script is finished, it will return the sidebar to the state it was when the script was launched.

Follow these steps to find the name of other buttons that reside in the object display area of a fixed page:

- Navigate to the page in question.
- Right click in the body of the page
- Select **Inspect** from the pop-up menu.
- In the search look for the text "Widget Info". Note that if there are multiple sections on the page, you will find multiple "Widget Info" results. Based on the details shown, you are able to confirm which section you are looking for.
- Iconized buttons (e.g., search buttons) are represented as HTML images and their field names are prefixed with **IM**. The following is an example of the HTML on the Algorithm - Main page (notice the **IM** fields for the iconized buttons).

```
* Widget Info:  -
*     Widget_ID -, Element Type -- label info -- label -
*     ALG_CD, IT -- CI_ALG$ALG_CD --  Algorithm Code
*     IM_ALG_CD, IM -- $SEARCH_FOR_ALG_LBL --  Search for Algorithm
*     OWNER_FLG, IS -- $OWNER_FLG --  Owner
*     DESCR50, IT -- $DESCR50 --  Description
*     IM_Main_algtypCont, IM -- $F1_SHOW_ALG_CTX --  Show Algorithm Context Menu
*     ALG_TYPE_CD, IT -- $ALG_TYPE_CD --  Algorithm Type
*     IM_ALG_TYPE_CD, IM -- $SEARCH_ALGTYPE_LBL --  Search for Algorithm Type
```

- Transaction-specific actions buttons are represented as switches. The following is an example of the HTML on the Batch Job Submission - Main page (notice the **SW** fields for the buttons). Note, if you want to **Set focus** to such a field, you would move a **Predefined Value** of **TRUE** to the switch.

```
*     ACTION_Q_SW, BU -- $ACTION_Q_SW --  Duplicate & Queue
*     ACTION_CANCEL_SW, BU -- $CANCEL_SW --  Cancel
```

How To Substitute Variables In Text

You can substitute field values into a step's text string. You do this by prefixing the field name whose value should be substituted in the string with a **%**. For example, the message, "On **%COMPLETION_DTTM** this bill was completed, it's ending balance was **%ENDING_BALANCE**" contains two substitution variables (the bill's completion date / time and the bill's ending balance).

To substitute the value of an element from a data area you need to reference its XPath location as follows: **%=XPath=%**. If you want to substitute the whole XML node, not just the value, you need to reference it as follows **%+XPath+%**.

Only fields linked to the current To Do and fields that reside in temporary storage and global variables can be substituted into a text string.

Note: You can substitute fields that reside in the User Interface or Page Data Model by first moving them into temporary storage (using a **Move data** step).

You can also substitute field values into the verbiage displayed in prompts using the same technique.

Copyright © 2024, Oracle and/or its affiliates.

How To Use HTML Tags And Spans In Text Strings and Prompts

You can use HTML tags in a step's text string. For example, the word "Continue" will be italicized in the following text string "PressContinue after you've selected the customer" (the **** and **** are the HTML tags used to indicate that the surrounded text should be italicized).

The following are other useful HTML tags:

- **
** causes a line break in a text string. If you use **

** a blank line will appear.
- **** text **** causes the surrounded text to be colored as specified (in this case, red). You can also use hex codes rather than the color name.

Please refer to an HTML reference manual or website for more examples.

Note: Refer to Color Contrast for information about the use of the HTML color 'red' and its impact on accessibility.

How To Use Constants In Scripts

Some steps can reference fields called **Predefined Values**. For example, if you want to compare an input value to the letter "Y", the letter **Y** would be defined as a Predefined Value's field value.

Special constants are used for fields defined as switches. When you move **TRUE** to a switch, it turns it on. When you move **FALSE** to a switch, it turns it off.

You can use a global variable as a Predefined Value. For example, if you wanted to move the current date to a field, you'd indicate you wanted to move a Predefined Value named **%CURRENT_DATE**.

How To Use Global Variables

Some explicit steps can reference fields called **Predefined Values**. In addition to referencing an ad hoc constant value (e.g., the letter **Y**), you can also reference a global variable in such a field value. A global variable is used when you want to reference system data.

Note that when using the Edit Data step type, the variable available are slightly different. Refer to Edit Data Syntax for details.

The following global variables exist for BPA scripts:

Variable Name	Comments
%PARM-<name>	This is the value of a parameter of that name passed in to the application when launched via the standard system URL. Refer to Launching A Script When Starting the System for more information on these parameters.
%PARM-NOT-SET	This is to be used to compare against **%PARM-< >** parameters to check if the parameter has been set or not when the application was launched. A parameter that has not been set would be considered equal to this global variable. It is recommended to compare parameters against this global variable before using them for the first time.
%BLANK	A constant that contains a blank value (no value).
%SPACE	A constant that contains a single space value.
%CURRENT-DATE	The current date as known by the browser, not the server.
%SYSTEM-DATE	The server date. Note that this date is affected by the system date override logic)
%SAVE-REQUIRED	A flag that contains an indication of whether the data on a page has been changed (and thus requires saving). You may want to interrogate this flag to force a user to save their work before executing subsequent steps. This flag will have a value of **TRUE** or **FALSE**.

Copyright © 2024, Oracle and/or its affiliates.

Variable Name	Comments
%NEWLINE	A constant that contains a new line character (carriage return). Upon substitution, a line break is inserted in the resultant text.
	Note: This constant does not have the desired effect when the resultant text is HTML. For example, a step's text and prompt strings. This is because HTML ignores special characters such as new lines. Refer to How To Use HTML Tags And Spans In Text to learn how to cause a line break in an HTML text.

To refer to a global context variable, use %FIELD_NAME. For example, if the field SP_ID is in the global context, you may reference %SP_ID to reference the ID of the service point currently in context. In addition, the following special values are supported:

Variable Name	Comments
%CONTEXT-PERSONID	A constant that contains the ID of the current person.
%CONTEXT-ACCOUNTID	A constant that contains the ID of the current account.
%CONTEXT-PREMISEID	A constant that contains the ID of the current premise.

In addition, if the script is invoking something else via one of the various “Invoke” step types and an error is returned, the following global variables contain information about the error:

Variable Name	Comments
%ERRMSG-CATEGORY **%ERRMSG-NUMBER**	The unique identifier of the error message number.
%ERRMSG-TEXT	The brief description of the error.
%ERRMSG-LONG	The complete description of the error.

How To Name Temporary Storage Fields

Input Data and **Move Data** steps can create fields in temporary storage. You specify the name of the temporary storage field in the step's **Field Name**. The name of the field must not begin with % and must not be named the same as the global variables. Besides this restriction, you can use any **Field Name** that's acceptable to JavaScript (i.e., you can name a field in temporary storage almost anything). Keep in mind that field names are case-sensitive.

How To Work With Dates

Before we discuss how to work with dates in your scripts, we need to point out that there are two types of date fields: date-only and date-time. Date-only fields only contain a date. Date-time fields contain both a date and a time. The following topics describe how to work with dates on the various step types.

Note: If you're working with a field that resides on the database (as opposed to a temporary storage field), the database field name will tell you what type of date it is: date-only fields are suffixed with **DT**, and date-time fields are suffixed with **DTTM**.

Move Data

If you intend to use a **Move data** step to populate a *date-time* field, please be aware of the following:

- If the destination field resides in the *page data model*, the source field value must be in the format YYYY-MM-DD-HH.MM.SS or YYYY-MM-DD. If the field is in the format YYYY-MM-DD, the time of 12:00 am will be defaulted.
- If the destination field resides in the *user interface*, you must use two steps if you want to populate both date and time. To explain this, we'll assume the field you want to populate is called EXPIRE_DTTM:

Copyright © 2024, Oracle and/or its affiliates.

- First, you populate the date portion of the field. To do this, you'd move a date (this value can be in any valid date format that a user is allowed to enter) to a field called EXPIRE_DTTM_FWDDTM_P1. In other words, you suffix **_FWDDTM_P1** to the field name.
- If you want to populate the time, you'd move the time (again, the field value can be in any format that a user could use to enter a time) to a field called EXPIRE_DTTM_FWDTTM_P2. In other words, you suffix **_FWDDTM_P2** to the field name.

If you intend to use a **Move data** step to populate a *date-only* field, please be aware of the following:

- If the destination field resides in the *page data model*, the source field value must be in the format YYYY-MM-DD.
- If the destination field resides in the *user interface*, the source field can be in any valid date format that a user is allowed to enter.

Note: %CURRENT-DATE. Keep in mind that the global variable%CURRENT-DATE contains the current date and you can move this to either a page data model, user interface, or temporary storage field. If you move %CURRENT-DATE to a temporary storage fields, it is held in the format YYYY-MM-DD.

Mathematical Operation

If you intend to use a **Mathematical operation** step to calculate a date, you can reference both date-only and date-time fields. This is because mathematical operations are only performed against the date portion of date-time fields.

Mathematical operations are limited to adding or subtracting days, months and years to / from a date.

Note: A useful technique to perform date arithmetic using the current date is to move the global variable%CURRENT-DATE to a temporary storage field and then perform the math on this field.

Input Data

If you intend to use an **Input data** step on a *date-time* field, please be aware of the following:

- If the field resides in the *page data model*, the user must enter a value in the format YYYY-MM-DD-HH.MM.SS (and therefore we do not recommend doing this).
- If the field resides in the *user interface*, you must use two steps if you want to populate both date and time. To explain this, we'll assume the field you want to populate is called EXPIRE_DTTM:
  - First, you populate the date portion of the field. To do this, you'd input the date (this value can be in any valid date format that a user is allowed to enter) in a field called EXPIRE_DTTM_FWDDTM_P1. In other words, you suffix **_FWDDTM_P1** to the field name.
  - If you want to populate the time, you'd input the time (again, the field value can be in any format that a user could use to enter a time) in a field called EXPIRE_DTTM_FWDTTM_P2. In other words, you suffix **_FWDDTM_P2** to the field name.

If you intend to use an **Input data** step to populate a *date-only* field, please be aware of the following:

- If the field resides in the *page data model*, the user must enter a value in the format YYYY-MM-DD (and therefore we do not recommend doing this).
- If the field resides in the *user interface*, the user can enter any valid date format.

How To Use To Do Fields

As described under Executing A Script When A To Do Entry Is Selected, you can set up the system to automatically launch a script when a user selects a To Do entry. These types of scripts invariably need to access data that resides on the selected To Do entry. The following points describe the type of information that resides on To Do entries:

- **Sort keys**. These values define the various ways a To Do list's entries may be sorted. For example, when you look at the bill segment error To Do List, you have the option of sorting the entries in error number order, account name order, or in customer class order. There is a sort key value for each of these options.
- **Message parameters**. These values are used when the system finds *%n* notation within the message text. The *%n* notation causes field values to be substituted into a message before it's displayed. For example, the message text **The %1 non-cash deposit for %2 expires on %3** will have the values of three fields merged into it before it is displayed

Copyright © 2024, Oracle and/or its affiliates.

to the user (**%1** is the type of non-cash deposit, **%2** is the name of the customer, and **%3** is the expiration date of the non-cash deposit). Each of these three values is stored as a separate message parameter on the To Do entry.

- **Drill keys**. These values are the keys passed to the page if a user drilled down on the entry (and the system wasn't set up to launch a script). For example, a To Do entry that has been set up to display an account on the account maintenance page has a drill key of the respective account ID.
- **To Do ID**. Every To Do entry has a unique identifier referred to as its To Do ID.

You can access this information in the following types of steps:

- **Move Data** steps can move any of the above to any data area. For example, you might want to move a To Do entry's drill key to the page data model so it can be used to navigate to a specific page.
- **Conditional Branch** steps can perform conditional logic based on any of the above. For example, you can perform conditional logic based on a To Do entry's message number (note, message numbers are frequently held in sort keys).
- **Mathematical Operation** steps can use the above in mathematical operations.

A To Do entry's sort key values are accessed by using a **Field Type** of **Current To Do Information** and a **Field Name** of **SORTKEY[index]**. Note, you can find an entry's potential sort keys by displaying the entry's To Do type and navigating to the Sort Keys tab. If you want to reference the first sort key, use an index value of **1**. If you want to use the second sort key, use an index value of **2** (and so on).

A To Do entry's drill key values are accessed by using a **Field Type** of **Current To Do Information** and a **Field Name** of **DRILLKEY[index]**. Note, you can find an entry's potential drill keys by displaying the entry's To Do type and navigating to the Drill Keys tab. If you want to use the first drill key, use an index value of **1**. If you want to use the second drill key, use an index value of **2** (and so on).

A To Do entry's message parameters are accessed by using a **Field Type** of **Current To Do Information** and a **Field Value** of **MSGPARM[index]**. Note, because a To Do type can have an unlimited number of messages and each message can have different parameters, finding an entry's message parameters requires some digging. The easiest way to determine these values is to display the To Do entry on To Do maintenance. On this page, you will find the entry's message category/number adjacent to the description. Once you know these values, display the message category/number on Message Maintenance. You'll find the message typically contains one or more %n notations (one for each message parameter). For example, the message text **The %1 non-cash deposit for %2 expires on %3** has three message parameters. You then need to deduce what each of the message parameters are. You do this by comparing the message on the To Do entry with the base message (it should be fairly intuitive as to what each message parameter is). If we continue using our example, **%1** is the non-cash deposit type, **%2** is the account name, and **%3** is the expiration date. You can access these in your scripts by using appropriate index value in **MSGPARM[index]**.

A To Do entry's unique ID is accessed by using a **Field Type** of **Current To Do Information** and a **Field Value** of **TD_ENTRY_ID**.

In addition, any of the above fields can be substituted into a text string or prompt. Simply prefix the To Do field name with a % as you would fields in temporary storage. For example, assume you want your script to display the following text in the script area: "ABC Supply does not have a bill cycle" (where ABC Supply is the account's name). If the first sort key linked to the To Do entry contains the account's name, you'd enter a text string of **%SORTKEY[1] does not have a bill cycle**.

How To Reference Fields In Data Areas

Various step types involve referencing field elements residing in the script's data areas. To reference an element in a data area you need to provide its absolute XPath notation starting from the data area name. For example, use "F1-BatchControl/batchControl" to reference a top-level "batchControl" element in a script data area called "F1–BatchControl".

You don't have to type in long XPath notations. Use the **View Script Schema** hyperlink provided on the Script - Step tab page to launch the script's data areas schema.

Copyright © 2024, Oracle and/or its affiliates.

Figure 4: Schema Viewer

Doing this opens the schema viewer window where you can:

- Click on the field element you want to reference in your script step. The system automatically populates the text box on the top with the element's absolute XPath notation.
- Copy the element's XPath notation from the text box to your script.

You can also use the **View Data Area**, **View Service Script Data Area**, or **View Plug-In Script Data Area** links on Script - Data Area to the same effect. These open up the schema viewer for a specific data area respectively.

Script Engine Version 2 and Above Notes

Scripting using the engine version 2 or above requires some extra syntax to take advantage of XPath 2 functionality. In general, any variable declared will be assumed to be a string. This means, that if you intend to construct a mathematical statement then it is necessary to explicitly declare the data type of variables as integers, numbers, or dates.

Note: Unless otherwise noted, all XPath examples in this topic are for the Version 1 engine – which means XPath 1. Statements that function using XPath 1 will not necessarily work for XPath 2. This is especially true when executing math, see below for examples.

Date and Time Arithmetic

XPath date/time and interval data types support arithmetic operations ('+', '-', '*' etc.) and functions, which can be used for time calculations in the same way as '1 + xs:integer(value)' is used for numeric calculations.

Compare time duration:

```
if ("(xs:dateTime(fn:current-dateTime()) -- xs:dateTime($updateDateTimeX))
   ge xs:dayTimeDuration(concat('PT', BO/hoursBetweenStatisticsUpdate, -'H'))")
   goto 60;
end-if;
```

Compare one date to another:

```
if ("xs:date(parm/endDate) < xs:date(parm/startDate)")
   terminate with error (11108, 11507 element='endDate');
end-if;
```

Compare a date against today's date:

```
if ("xs:date(parm/startDate) <= xs:date($CURRENT-DATE)")
   terminate with error (11108, 11507 element='endDate');
end-if;
```

Calculate the end of month:

```
// covert to ISO
move -"concat($year,'-',$mon2,'-01T00:00:00')" to $monthStart; -
 -
// calculate
move -"xs:dateTime($monthStart) + xs:yearMonthDuration('P1M') -- -
xs:dayTimeDuration('P0DT1S')" -
  to $monthEnd;
 -
// convert from ISO to OUAF
```

Copyright © 2024, Oracle and/or its affiliates.

```
move -"concat($year,'-',$mon2,'-',substring(string($monthEnd),9,2),'-23.59.59')" to -
$endDateTime;
```

Note: XPath date/time/interval formats use the ISO standard, which needs to be converted to/from formats supported in the framework.

Comparing Date/Times in String Format

Any ISO-like string format for date/time preserves the YYYY MM DD HH MM SS sequence, which is zero-padded. Regardless of separators, this format will remain appropriate for comparison operations. In particular, date/time values in the framework format “YYYY-MM-DD.HH.MM.SS” can be used with “=”, “!=”, as well as “>”, “>=”, “<”, “<=” operators.

```
// retrieve framework date/time value
invokeBS -'CM-MAXMSRMT' using -"CM-MAXMSRMT";
move -"string(cm-MAXMSRMT/results[1]/measurementDateTime)" to $lastMsmtDT;
 -
// construct another date/time
move -"concat($year,'-01-01-00.00.00')" to $startDateTime;
 -
// compare using string operators
if ("$lastMsmtDT >= $startDateTime")
    move -"substring($lastMsmtDT,1,4)" to $latestMsrmtYear;
```

Converting Date/Times Between Framework and ISO

Conversion of date/time from framework format to ISO is only necessary for date/time arithmetic. Comparisons can be done with the framework format directly. The only difference between the framework format and ISO date/time formats is in the separators:

Framework: “YYYY-MM-DD.HH.MM.SS”

ISO: “YYYY-MM-DDTHH:MM:SS”

Example of converting from the framework format to ISO:

```
move -"concat(substring($ouafDT, 1, 10), -'T', translate(substring($ouafDT, -
12),'.',':'))" to $isoDT;
```

Example of converting from ISO to the framework format:

```
move -"concat(substring($isoDT, 1, 10), -'.', translate(substring($isoDT, -
12),':','.'))" to $ouafDT;
```

Round Money With a Dynamic Currency Scale

Because different currencies support a different number of decimals, the framework provides an API for rounding a monetary amount based on a given currency.

```
move -"parm/amount" to $qnty;
move -"currency/decimals" to $decimals;
move -"fn:round(xs:decimal($qnty) * math:exp10(xs:double($decimals))) -
  div math:exp10(xs:double($decimals))" to -"parm/roundedAmount";
```

Looping through Sequences

In XPath 2 it is possible to organize a for-loop over a sequence of integers, not only a node list.

Copyright © 2024, Oracle and/or its affiliates.

This example shows a loop over a range of months. This is a sequence-forming construct in XPath. The XPath node list, which we are familiar with, is just another type of sequence.

```
for ($month in -"1 to 12")
```

This example shows a loop over a give range of years in descending order:

```
for ($year in -"fn:reverse(parm/startYear to parm/endYear)")
     move -"concat($year,'-01-01-00.00.00')" to $startDateTime;
     move -"concat($year,'-12-31-23.59.59')" to $endDateTime;
     -...
```

This example shows a loop through a node list using ‘index’, so that other node lists can be accessed:

```
for ($idx in -"1 to count(parm/touData/touList)")
     move -"parm/touData/touList[$idx]" to $tou;   -// access any list with this -
index
```

The above syntax can be used as an elegant alternative to maintaining indices separately, for example instead of the following:

```
move -“0” to $idx;
for ($item in -"parm/touData/touList")
     move -“1 + xs:integer($idx)” to $idx;
```

String Padding and Decimal Formatting

This is used with specific input formats or output formatting. It is applicable to zero, space and other types of padding.

This example shows prefixing for date/time components, for example producing “2010-01-02” instead of “2010-1-2”.

```
move -"substring(concat('0',string($month)), string-length(string($month)), 2)" to -
$mon2;
```

This example shows suffixing for adding decimal zero-padded alignment, for example producing “12.30” and “4.00” instead of “12.3” and “4”. The example performs 3 tasks: rounding to 2 decimals, inserting a period if necessary, and zero padding.

```
// round and zero-pad to 2 decimals
move -"$item/amount" to $qty;
move -"fn:round(xs:double($qty) * 100) div 100" to $qty;
move -"string($qty)" to $qty;
move -"concat(substring-before(concat($qty,'.'),'.'),'.',substring(concat(substring-
after($qty,'.'),'00'),1,2))" to $qty;
```

Ternary Operation

This makes a choice between values based on a condition, so that it could be used in a single expression instead of an if/else block. It is know in C/C++ as ‘cond ? value1 : value2’ or in BASIC as ‘IFF(cond, value1, value2)’. In XPath the syntax is: "if (cond) then value1 else value2". Note this is not the top-level scripting if-statement block.

In XPath this is an expression, which can be combined with other expressions. In scripting it can be used as:

```
move -"if (string(D1-UnitOfMeasure/measuresPeakQuantity) = -'D1MP') then -'D1MX' -
else -'D1SM' -" to $func;
```

Pipeline Processing

In scripting, it is not easy to create a simple reusable piece of code as there are no local functions, and a separate script call is a coding overhead and requires packing/unpacking parameters. To avoid copying and pasting the same code block between similar script stages, consider ‘pipelining’, which is breaking the overall process into separate top-level

Copyright © 2024, Oracle and/or its affiliates.

steps, some of which could be shared between alternating paths. This is common for parameter preparation and output formatting. An intermediate result between stages can be stored in a “parm” substructure.

Instead of this code:	Consider this alternative:

`if ("type = A")`
`  prepare params -...`
`  call services for A -...`
`  format output -...`
`end-if;`
`if ("type = B")`
`  prepare params -...`
`  call services for B -...`
`  format output -...`
`end-if;`
	`prepare params -...`
	
	`if ("type = A")`
	`  call services for A -...`
	`end-if;`
	`if ("type = B")`
	`  call services for B -...`
	`end-if;`
	
	`format output -...`

XPath 2 Functions

Script engine versions 2 and above support XQuery 1.0 Functions and Operators, and the XQuery 1.0 standard itself with some minor limitations. Below are the URLs to both specifications. The first link has the functions/operators available to use from XQuery.

- http://www.w3.org/TR/xpath-functions/
- http://www.w3.org/TR/xquery/

The following can only access local file systems. (For other protocols like http they will return an empty sequence):

- fn:doc
- fn:collection

Script - Data Area

Use this page to define the data areas used to pass information to and from the server or any other data area describing your temporary storage. Open this page using **Admin** > **System** > **Script** and then navigate to the **Data Area** tab.

Note: Conditional tab page. This tab page does not appear for Groovy Library scripts or plug-in scripts using the **Groovy** engine version.

Description of Page

The grid contains the script's data areas declaration. For steps that invoke an object that is associated with a schema, you must declare the associated schema as a data area for your script. In addition, if you have defined one or more data areas to describe the script's temporary storage, you need to declare them too. The following bullets provide a brief description of each field on a script data area:

- **Schema Type** defines the type of schema describing the data area's element structure.
- The data area's schema is the one associated with the referenced **Object**. Only objects of the specified Schema Type may be selected.
- **Data Area Name** uniquely identifies the data area for referencing purposes. By default, the system assigns a data area with the associated object name.
- Click on the **View Data Area** link to view the data area's schema in the schema viewer window.

The **View Service Script Data Area** link appears for service scripts only. Use this link to view the script's parameters data area schema in the schema viewer window.

The **View Plug-In Script Data Area** link appears only for plug-in scripts using a script engine version. Use this link to view the script's parameters data area schema in the schema viewer window.

Fastpath: Refer to A Script May Declare Data Areas for more information on data areas.

Copyright © 2024, Oracle and/or its affiliates.

Script - Schema

Use this page to define the data elements passed to and from a service script. Open this page using **Admin** > **System** > **Script** and then navigate to the **Schema** tab.

Note: Conditional tab page. This tab page only appears for service scripts.

Description of Page

The contents of this section describe the zones that are available on this portal.

The **General Information** zone displays the script name and description.

The Schema Designer zone allows you to edit the service script's parameters schema. The purpose of the schema is to describe the input and output parameters used when invoking the script.

Note: Refer to Schema Nodes and Attributes for a complete list of the XML nodes and attributes available to you when you construct a schema.

The **Schema Usage Tree** zone summarizes all cross-references to this schema. For each type of referencing entity, the tree displays a summary node showing a total count of referencing items. The summary node appears if at least one referencing item exists. Expand the node to list the referencing items and use their description to navigate to their corresponding pages.

Script - Eligibility

Use this page to define a script's eligibility rules. Open this page using **Admin** > **System** > **Script** and then navigate to the **Eligibility** tab.

Note: Conditional tab page. This tab page only appears for BPA scripts.

Description of Page

Use the **Eligibility Option** to indicate whether the script is **Always Eligible**, **Never Eligible** or to **Apply Eligibility Criteria**. The remaining fields on the page are only visible if the option is **Apply Eligibility Criteria**.

CAUTION: The following information is not intuitive; we strongly recommend that you follow the guidelines under The Big Picture Of Script Eligibility before attempting to define this information.

The **Eligibility Criteria Group** scroll contains one entry for each group of eligibility criteria. The following fields may be defined for each group:

- Use **Sort Sequence** to control the relative order in which the group is executed when the system determines if the script should appear in the script search.
- Use **Description** and **Long Description** to describe the criteria group.
- Use **If Group is True** to define what should happen if the eligibility criteria (defined in the following grid) return a value of **True**.
  - Choose **Eligible** if this script should appear.
  - Choose **Ineligible** if this script should not appear.
  - Choose **Check Next Group** if the next criteria group should be checked.
- Use **If Group is False** to define what should happen if the eligibility criteria (defined in the following grid) return a value of **False**.
  - Choose **Eligible** if this script should appear.
  - Choose **Ineligible** if this script should not appear.
  - Choose **Check Next Group** if the next criteria group should be checked.

The grid that follows contains the script's eligibility criteria. Think of each row as an "if statement" that can result in the related eligibility group being true or false. For example, you might have a row that indicates the script is eligible if the current account in context belongs to the residential customer class. The following bullets provide a brief description

Copyright © 2024, Oracle and/or its affiliates.

of each field on an eligibility criterion. Please refer to Defining Logical Criteria for several examples of how this information can be used.

- Use **Sort Sequence** to control the order in which the criteria are checked.
- Use **Criteria Field** to define the field to compare:
  - Choose **Algorithm** if you want to compare anything other than a characteristic. Push the adjacent search button to select the algorithm that is responsible for retrieving the comparison value.
  - Some products may also include an option to choose **Characteristic**. Choosing this option displays adjacent fields to define the object on which the characteristic resides and the characteristic type. The objects whose characteristic values may be available to choose from depend on your product.
- Use **Criteria Comparison** to define the method of comparison:
  - Choose **Algorithm** if you want an algorithm to perform the comparison and return a value of True, False or Insufficient Data. Push the adjacent search button to select the algorithm that is responsible for performing the comparison.
  - Choose any other option if you want to compare the **Criteria Field** using a logical operator. The following options are available:
    - Use >, <, =, >=, <=, <> (not equal) to compare the **Criteria Field** using standard logical operators. Enter the comparison value in the adjacent field.
    - Use **IN** to compare the **Criteria Field** to a list of values. Each value is separated by a comma. For example, if a field value must equal **1**, **3** or **9**, you would enter a comparison value of **1,3,9**.
    - Use **BETWEEN** to compare the **Criteria Field** to a range of values. For example, if a field value must be between **1** and **9**, you would enter a comparison value of **1,9**. Note, the comparison is inclusive of the low and high values.
- The next three fields control whether the related logical criteria cause the eligibility group to be considered true or false:
  - Use **If True** to control what happens if the related logical criterion returns a value of True. You have the options of **Group is true**, **Group is false**, or **Check next condition**. If you indicate **Group is true** or **Group is false**, the script is judged **Ineligible** or **Eligible** based on the values defined above in **If Group is False** and **If Group is True**.
  - Use **If False** to control what happens if the related logical criterion returns a value of False. You have the options of **Group is true**, **Group is false**, or **Check next condition**. If you indicate **Group is true** or **Group is false**, the script is judged **Ineligible** or **Eligible** based on the values defined above in **If Group is False** and **If Group is True**.
  - Use **If Insufficient Data** to control what happens if the related logical criterion returns a value of "Insufficient Data". You have the options of **Group is true**, **Group is false**, or **Check next condition**. If you indicate **Group is true** or **Group is false**, the script is judged **Ineligible** or **Eligible** based on the values defined above in **If Group is False** and **If Group is True**.

Merging Scripts

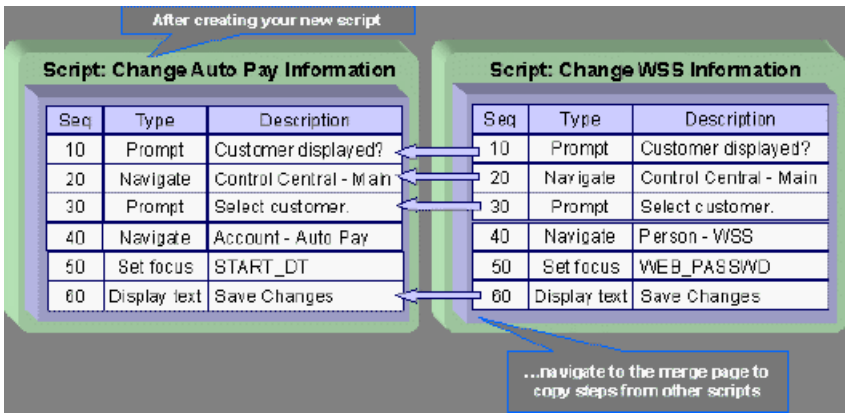
Use the Script Merge page to modify an existing script by copying steps from other scripts. The following points summarize the many diverse functions available on the Script Merge transaction:

- You can use this transaction to renumber steps (assign them new sequence numbers).
- You can use this transaction to move a step to a different position within a script. When a step is moved, all references to the step are changed to reflect the new sequence number.
- You can use this transaction to delete a step.
- You can use this transaction to copy steps from other scripts. For example:
  - You may want to create a script that is similar to an existing script. Rather than copying all the information from the existing script and then removing the inapplicable steps, this page may be used to selectively copy steps from the existing script to the new script.

Copyright © 2024, Oracle and/or its affiliates.

- You may have scripts that are very similar, but still unique. You can use this transaction to build large scripts from smaller scripts. In this scenario, you may choose to create special 'mini' scripts, one for each of the various options that may make a script unique. Then, you could use the script merge page to select and merge the mini scripts that are applicable for a main script.

Note: The target script must exist prior to using this page. If you are creating a new script, you must first create the Script and then navigate to the merge page to copy step information.



Note: Duplicate versus Merge. The Script page itself has duplication capability. You would duplicate a script if you want to a) create a new script and b) populate it with *all* the steps from an existing script.

Script Merge

Open **Admin** > **System** > **Script Merge** to open this page.

Description of Page

For **Original Script**, select the target script for merging steps.

For **Merge From Script**, select the template script from which to copy the steps.

Note: You may only copy steps from one Merge From script at a time. If you want to copy steps from more than one script, select the first Merge From script, copy the desired steps, save the original script, and then select the next Merge From script.

The left portion of the page displays any existing steps for the **Original Script**. The right portion of the page displays the existing steps for the **Merge From Script**.

You can use the **Copy All** button to copy all the steps from the **Merge From** script to the **Original** script. If you use **Copy All**, the steps are added to the end of the original script.

Each time you save the changes, the system renumbers the steps in the original script using the **Start From Sequence Number** and **Increment By**.

Merge Type indicates **Original** for steps that have already been saved in the original script or **Merge** for steps that have been merged, but not yet saved. The **Sequence**, **Step Type** and **Description** for each step are displayed.

The topics that follow describe how to perform common maintenance tasks:

Resequencing Steps

If you need to resequence the steps:

- Use the up and down arrows in the Original Script grid to reorder the steps.
- Make any desired changes to the **Start From Sequence Number** or **Increment By**.
- Click Save.

Copyright © 2024, Oracle and/or its affiliates.

The steps are given new sequence numbers according to their order in the grid.

Removing a Step from Script

If you want to remove a record linked to the Original script, click the delete button to the left of the record.

For example, to remove the **Reset existing bundle XML** step, click the icon.

		Merge Type	Sequence	Step Type	Description
		Original	10	Edit data	Edit data - Check that the BO is an Export Bundle
		Original	20	Edit data	Edit data - Read the Bundle
		Original	30	Edit data	Edit data - Reset existing bundle XML
		Original	40	Edit data	Edit data - Create new bundle XML

After removal, the grid displays:

		Merge Type	Sequence	Step Type	Description
		Original	10	Edit data	Edit data - Check that the BO is an Export Bundle
		Original	20	Edit data	Edit data - Read the Bundle
		Original	40	Edit data	Edit data - Create new bundle XML

Note: You cannot delete a step that is referenced by other steps unless you also delete the referencing steps, such as **Go to step** or **Prompt** type steps. The system informs you of any missing referenced steps when you attempt to save the original script.

Adding a Step to a Script

You can move any of the steps from the Merge From script to the Original Script by clicking the left arrow adjacent to the desired step. Once a record is moved it disappears from the Merge From information and appears in the Original information with the word **Merge** in the Merge Type column.

For example, to copy the **Navigate to a page** step, click the left arrow.

Copyright © 2024, Oracle and/or its affiliates.

	Merge Type	Sequence	Step Type	Description

	Sequence	Step Type	Description
	10	Height	Height - 0%
	20	Move data	Move data - Copy 'APP_SVC_ID' 'APP_SVC_ID'
	30	Move data	Move data - Copy 'USR_GRP_ID' 'USR_GRP_ID'
	40	Navigate to a page	Navigate to a page - userGroupAppService
	50	Conditional branch	Conditional branch - Compare A with UPDATE
	60	Press a button	Press a button - IM_scrollSec_ad
	70	Move data	Move data - Copy 'APP_SVC_ID' 'UGP$APP_SVC_ID'
	80	Label	Label - End of Script

The step is moved to the left portion of the page.

Merge Type	Sequence	Step Type	Description
Merge	40	Navigate to a page	Navigate to a page - userGroupAppService

	Sequence	Step Type	Description
	10	Height	Height - 0%
	20	Move data	Move data - Copy 'APP_SVC_ID'
	30	Move data	Move data - Copy 'USR_GRP_ID'
	50	Conditional branch	Conditional branch with UPDATE
	60	Press a button	Press a button - IM
	70	Move data	Move data - Copy 'UGP$APP_SVC_ID
	80	Label	Label - End of Scrip

Note: If you add a step, such as **Go to step** or **Prompt** type steps, that references other steps, you must also add the referenced steps. The step references are updated to use the new sequence numbers when you save the original script. The system informs you of any referenced steps that haven't been added when you attempt to save the original script.

Copyright © 2024, Oracle and/or its affiliates.

Removing an Uncommitted Step from a Script

Merge Type	Sequence	Step Type	Description
Merge	40	Navigate to a page	Navigate to a page - userGroupAppService

d Step - Script Merge - row 1

	Sequence	Step Type	Description
	10	Height	Height - 0%
	20	Move data	Move data - Copy ' 'APP_SVC_ID'
	30	Move data	Move data - Copy ' 'USR_GRP_ID'
	50	Conditional branch	Conditional branch with UPDATE
	60	Press a button	Press a button - IM
	70	Move data	Move data - Copy ' 'UGP$APP_SVC_ID
	80	Label	Label - End of Scri

Maintaining Functions

Note: Functions were implemented prior to the introduction of business services (BS), service scripts (SS) and business objects (BO). The functionality is still supported, but the recommendation for implementations going forward is to use one of the above configuration tool objects in a script rather than defining a function. The documentation has not been updated throughout this section to highlight where BS, SS or BO could be used to perform the equivalent logic.

Invoke function steps may be used to retrieve or update data independent of the page currently being displayed. For example, if you design a script that takes different paths based on the customer's customer class, you could invoke a function to retrieve the customer's customer class. Doing this is much more efficient than the alternative of transferring to the account page and retrieving the customer class from the Main page.

An **Invoke function** step retrieves or updates the relevant data by executing a service (on the server). These types of steps do not refer to the service directly. Rather, they reference a "function" and the function, in turn, references the service.

Note: Functions are abstractions of services. A function is nothing more than meta-data defining the name of a service and how to send data to it and retrieve data from it. Functions allow you to define a scriptwriter's interface to services. They also allow you to simplify a scriptwriter's set up burden as functions can handle the movement of data into and out of the service's XML document.

The topics in this section describe how to set up a function.

Note: You can retrieve data from all base-package objects. If you know the name of the base-package "page" service used to inquire upon an object, you can retrieve the value of any of its fields for use in your scripts. To do this, set up a function that sends the unique identifier of the object to the service and retrieves the desired fields from it.

Function - Main

Use this page to define basic information about a function. Open this page using **Admin** > **System** > **Function.**

Description of Page

Enter a unique **Function** code and **Description** for the function.

Use the **Long Description** to describe, in detail, what the function does.

Define the **Internal Service** that the function invokes.

Copyright © 2024, Oracle and/or its affiliates.

Note: Only page services can be invoked.

The tree summarizes the following:

- The fields sent to the service. You can use the hyperlink to transfer to the **Send Fields** tab with the corresponding field displayed.
- The fields received from the service. You can use the hyperlink to transfer to the **Receive Fields** tab with the corresponding field displayed.
- Scripts that reference the function. You can use the hyperlink to transfer to the script page.

Function - Send Fields

Use this page to add or update the fields sent to the service. Open this page using **Admin** > **System** > **Function** and then navigate to the **Send Fields** tab.

Note: Displaying a specific field. Rather than scrolling through each field, you can navigate to a field by clicking on the respective node in the tree on the Main tab. Also note, you can use the Alt+right arrow and Alt+left arrow accelerator keys to quickly display the next and previous entry in the scroll.

Note: You're defining the service's input fields. On this tab, you define which fields are populated in the XML document that is sent to the service. Essentially, these are the service's input fields.

Description of Page

Use **Sequence** to define the order of the **Send Fields**.

Enter a unique **Function Field Name** and **Description** for each field sent to the application service. Feel free to enter **Comments** to describe how the field is used by the service.

Use **Field Value Source** to define the source of the field value in the XML document sent to the service:

- If the field's value is the same every time the function is invoked, select **Defined On The Function**. Fields of this type typically are used to support "hard-coded" input values (so that the scriptwriter doesn't have to populate the field every time they invoke the function). Enter the "hard-coded" **Field Value** in the adjacent field.
- If the field's value is supplied by the script, select **Supplied By The Invoker**. For example, if the function retrieves an account's customer class, the script would need to supply the value of the account ID (because a different account ID is passed each time the function is invoked). Turn on **Required** if the invoker must supply the field's value (it's possible to have optional input fields).

Regardless of the Field Value Source, use **XML Population Logic** to define the XPath expression used to populate the field's value in the XML document sent to the service.

Function - Receive Fields

Use this page to add or update the fields received from the service. Open this page using **Admin** > **System** > **Function** and then navigate to the **Receive Fields** tab.

Note: Displaying a specific field. Rather than scrolling through each field, you can navigate to a field by clicking on the respective node in the tree on the Main tab. Also note, you can use the Alt+right arrow and Alt+left arrow accelerator keys to quickly display the next and previous entry in the scroll.

Note: You're defining the application service's output fields. On this tab, you define which fields are populated in the XML document that is received from the service. Essentially, these are the service's output fields.

Description of Page

Use **Sequence** to define the order of the **Receive Fields**.

Enter a unique **Function Field Name** and **Description** for each field received from the service. Feel free to enter **Comments** to describe the potential values returned from the service.

Copyright © 2024, Oracle and/or its affiliates.

Turn on **Required** if the invoker must use the field.

Regardless of the Field Value Source, use **XML Population Logic** to define the XPath expression used to retrieve the field's value from the XML document received from the service.

> **Note: Fields in multiple lists.** If a field within the list is unique across the service, the generic list reference is sufficient for the XML population logic. However, if the field you are trying to reference is in multiple lists, the XPath must include the list name. For example, instead of **/pageBody/list/listBody/field[@name='FIELD_NAME']**, the XPath Population Logic must read **/pageBody/list[@name='LIST_NAME']/listBody/field[@name='FIELD_NAME']**.

Maintaining Managed Content

The Managed Content object is used to store various types of text content. Some content is used internally such as Allowlists. Other content may be used by implementations such as XSL files used for communicating messages with an external system, JavaScript include files, and CSS files. These files may then be maintained in the same manner as the HTML in UI Maps.

Open this page using **Admin** > **System** > **Managed Content**.

Use the **Managed Content Query** portal to search for an existing managed content entry. Once the managed content is selected, you are brought to the maintenance page to view and maintain the selected record.

Most fields on the page are self explanatory.

Use **Managed Content Type** to indicate the type of content.

- **Allowlist**

  is used to capture a list of supported values for given functionality. For example, the product provides allowlists for SQL functions and valid HTML tags.
- **XSLTC** is used to reference XSL files. These may be used internally for user interface referencing. Additionally, the XSLs may be referenced for sending messages to external system using outbound message and for receiving messages in SOAP as inbound messages.

> **Note:** The value F1X (XSLT) is no longer supported and will be deprecated in a future release. Other managed content values are not currently supported by any current framework functionality and are reserved for future use.

Use the **Schema** tab to view and maintain the contents of the managed context text.

Defining Entity Tags

This portal is used to maintain an entity tag along with the list of entities associated with it. Entity tagging supports grouping of configuration, demo and test related entities so they can better describe a logical function. Each group is represented by a unique tag name and entities may be associated with multiple tags.

An entity tag may explicitly restrict the type of entities it can be associated with by referencing a specific maintenance object as part of the entity tag record.

You can access the portal from the**Admin Menu** > **Implementation Tools** > **Entity Tag**. You are brought to a query portal with options for searching for a specific tag name. Once an entity tag has been selected you are brought to the maintenance portal to view and maintain the selected record.

A Tags sidebar zone allows you to review, add and remove tags for the current entity you review. For configuration type of entities, you may use this portal to tag multiple entities at once.

The following zones may appear as part of the portal's **Main** tab page

- **Entity Tag.** This zone provides information about the selected entity tag.

Copyright © 2024, Oracle and/or its affiliates.

- **Tagged Entities.** This zone lists entities associated with the selected entity tag. You may also use the zone to remove entities from the list.
- **Tag Entities.** Use this zone to add configuration entities to the selected tag.

Advanced Schema Topics

The topics in this section describe some advanced information related to schemas used for business objects, business services, service scripts and UI maps.

Schema Nodes and Attributes

For business object definition, the purpose of the schema is to create a link between the schema and a maintenance object. For business service definition you are specifying the link between the schema and a service (either a general service, search service, or a maintenance object service). For service script definition, you are defining the API for passing information to and from the script. The following documentation is a complete list of the XML nodes and attributes available to you when you construct a schema.

The Four Element Types

A schema element can be one of four different structure types. Note that there are two classes of element types: the structural nodes group and list, and the data containing nodes of field and raw.

Mnemonic	Valid Values	Description	Examples
type=	**"field"**	The field type is the default type for any element not explicitly labeled as something other than a field. Therefore, you virtually never have to explicitly label an element as a field. Note that a field element, unlike group or list, will contain information in its nodes - rather than other nodes.	

	"group"	The group element is typically a structural element of the schema only, in which case it has no mapping.
		
		Note that when grouping several elements that are all used to map an XML structure of a CLOB / XML field of a business object driven record, the mapping may be at the group level.
			Example where a group is used to create a structure
			
			`<schema>`
			`  <input type="group"`
			`  <userId/>`
			`  </input>`
			`  <output type="group">`
			`  <firstName/>`
			`  <lastName/>`
			`  </output>`
			`</schema>`
			
			Example where the group includes the mapping:
			
			`<parameters type="group" -`
			`mapXML="BO_DATA_AREA" mdField=`
			`"F1_TODOSUMEMAIL_PARM_LBL"/>`
			`  <numberOfDays -`
			`  mdField="F1_NBR_DAYS" -`
			`  required="true"/>`

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Valid Values	Description	Examples
			`<frequency - mdField="F1_FREQUENCY"/>`

	"list"	The list node is structural node like the group node. The only difference is that the list structure has the ability to repeat multiple times in an XML document.	Example of a schema with a list:
			`<schema>`
			`  <statesList type="list">`
			`  <state isPrimeKey="true"/>`
			`  <description/>`
			`  </statesList>`
			`</schema>`
			Example of a schema with a list:
			`<xml>`
			`  <statesList>`
			`  <state>AK</state>`
			`  <description>Alaska</description>`
			`  </statesList>`
			`  <statesList>`
			`  <state>AL</state>`
			`  <description>Alabama</description>`
			`  </statesList>`
			`...`
			`</xml>`

	"raw"	The raw data type is used to capture a chunk of raw text that doesn't have any inherent structure associated with it.	`<sendDetail type="raw" -/>`
			Example of an XML instance for the above schema:
			`<sendDetail>`
			`  <messageInfo>`
			`  <senderAddress>123 W. - Main St,  Ontario, CA`
			`  </senderAddress>`
			`  <corpZone>3A</corpZone>`
			`  </messageInfo>`
			`</sendDetail>`

The Data Type of a Field Element

Of the four different element types, only a field can have a data type.

Mnemonic	Valid Values	Description	Examples

dataType=	**"string"**	By default, a field element is a string. Therefore, there is no requirement to specify the string data type.	`<schema>`
			`  <custName dataType="string"/>`
			`</schema>`

	"number"	Defines an element that is a number.
		Note: UI hints include a setting to Suppress Automatic Number Formatting.
			Examples
			`<schema>`
			`  <count dataType="number"/>`
			`</schema>`
			`<schema>`
			`  <taxRate dataType="number" - currencyRef="currency"/>`

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Valid Values	Description	Examples
		Note: Use **currencyRef** attribute for auto-display of currency symbol that is associated with the referenced currency code. The currency decimal positions are ignored by this formatting allowing you to display a currency symbol for a unit rate with many decimals.	`</schema>`

	"money"
	Optional additional attributes
	currencyRef="element name"
		Defines an element that represents a monetary amount.
		The currency reference is optional and if left blank the installation currency will be used. Automatic formatting and validation to be applied based on the currency. For example, the currency symbol will be shown when auto-rendering. In addition, the number of decimal places must not exceed the valid number defined for the currency.
		Note: Refer to Referencing Other Elements for supported syntax for referring to other elements.
			`<schema>`
			`  <currency default="USD" -`
			`  suppress="true"/>`
			`  <balance dataType="money" -`
			`  currencyRef="currency"/>`
			`</schema>`

	"lookup"
	Required additional attribute
	lookup="field name"
		Defines an element that has valid values defined using a lookup. The lookup field is required.	`<schema>`
			`  <status dataType="lookup" -`
			`  lookup="STATUS_FLG"/>`
			`</schema>`

	"lookupBO"
	Required additional attribute
	lookupBO="bo name"
		Defines an element that has valid values defined using an extendable lookup. The extendable lookup's business object is required.	`<schema>`
			`  <category -`
			`dataType="lookupBO" -`
			`  lookupBO="CM-`
			`BusinessCategory"/>`
			`</schema>`

	"boolean"	Defines an element that has values of "Y" and "N".	`<schema>`
			`  <allowsEdit -`
			`dataType="boolean"/>`
			`</schema>`

	"date"	Defines an element that represents a date.	`<schema>` `  <startDate dataType="date"/>`

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Valid Values	Description	Examples
			`</schema>`

	"dateTime"	Defines an element that represents a date and time.
		Note: Refer to Standard Time Considerations for additional configuration available for date / time fields that represent standard time.
			`<schema>`
			`  <startDateTime -`
			`  dataType="dateTime"/>`
			`</schema>`

	"time"	Defines an element that represents a time.	`<schema>`
			`  <startTime dataType="time"/>`
			`</schema>`

	"uri"	Defines an element captures a URI. Elements defined with this type enable the support for URI Allowlist and Substitution as described in Referencing URIs .	`<schema>`
			`  <exportDirectory -`
			`dataType="uri"/>`
			`</schema>`

Referencing Other Elements

There are several attributes that allow for a reference to another element in the same schema. The supported syntax of the XPath reference is the same in every case. This section provides examples using the default reference attribute (**defaultRef**).

Reference a sibling element:

```
<schema>
   <id mapField="ACCT_ID" required="true"/>
   <altId defaultRef="id" required="true"/>
</schema>
```

Reference an element in a higher group:

```
<schema>
   <id mapField="ACCT_ID" required="true"/>
   <msgInfo type="group" mapXML="XML_FIELD">
     <altId defaultRef="../id" required="true"/>
   </msgInfo>
</schema>
```

Reference an element in a lower group:

```
<schema>
   <id mapField="ACCT_ID" defaultRef="msgInfo/altId" required="true"/>
   <msgInfo type="group" mapXML="XML_FIELD">
     <altId required="true"/>
   </msgInfo>
</schema>
```

Reference an element in another group:

```
<schema>
  <acctInfo type="group">
    <id mapField="ACCT_ID" required="true"/>
  </acctInfo>
   <msgInfo type="group" mapXML="XML_FIELD">
     <altId defaultRef="../acctInfo/altId" required="true"/>
```

Copyright © 2024, Oracle and/or its affiliates.

```
    </msgInfo>
</schema>
```

Standard Time Considerations

Most date / time fields represent "legal" time such that if a time zone changes their clocks for winter and summer time, the date / time field captures the current observed time. However, some date / time fields should always be captured in standard time to avoid confusion / ambiguity. A good example is a date and time related to detailed interval data. Refer to Designing Time Zones for more information.

When defining an element with **dataType="dateTime"**, you may optionally configure **stdTime="true"** indicating that data captured in the element always represents standard time in the 'base' time zone. The 'base' time zone is specified on the Installation options.

Note: If an element is mapped to a table / field with a Standard Time Type of **Physical**, then **stdTime="true"** is implied. Refer to Fields for more information.

Example:

```
<schema>
  <startTime dataType="Time" stdTime="true"/>
</schema>
```

If the time zone that represents the date / time field is not the installation time zone, use the optional setting **stdTimeRef=**"XPath to time zone element" on a date / time element to indicate that the element represents standard time and indicates the time zone to use. Refer to Referencing Other Elements for supported syntax for referring to other elements. .

Example:

```
<schema>
  <alternateTimeZone fkRef="F1-TZONE" suppress="true"/>
  <startDateTime dataType="dateTime" stdTimeRef="alternateTimeZone"/>
</schema>
```

Note: If an element is mapped to a table / field with a Standard Time Type of **Referenced**, then **stdTime=**"XPath" is implied. Refer to Fields for more information.

Note: When schema elements are captured in standard time the UI map supports HTML notation to automatically display the data applying a daylight savings time / summer time correction. Refer to the HTML attribute oraType="dateTime; stdTime:true" for more information.

There may be cases where the date / time is captured as standard time in one time zone, but should be displayed using a different time zone. In this case, the attribute **displayRef=**"XPath" may be used in addition to the appropriate attribute that identifies the time zone that the data is capture in. Refer to Referencing Other Elements for supported syntax.

```
<schema>
  <displayTimeZone fkRef="F1-TZONE" suppress="true"/>
  <startDateTime dataType="dateTime" stdTime="true" displayRef="displayTimeZone"/>
</schema>
```

The Mapping Attributes

When constructing your schema, you can choose from one of the following mapping attributes.

Mnemonic	Valid Values	Description	Examples

mapField=	"field name"	In the case of a business object, the **mapField** attribute is used to identify the database column the element is related to. For business	`<schema>`
			`  <factId -`
			`  mapField="FACT_ID"/>`
			`</schema>`

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Valid Values	Description	Examples
		service schemas, the **mapField=** attribute is used to link a schema element with a service element.	

mapChild=	"table name"	The **mapChild** attribute is used only for business object mapping. It is used in two ways:
		• First, to create a list in the business object that corresponds to a child table of an MO.
		• Second, you can use **mapChild** to identify the child table a flattened field lives in. For more information on flattening, refer to flattening section below.
			Example of a list within a child table in a BO:
			`<persons type="list" -`
			`  mapChild="CI_ACCT_PER"`
			`  <personId mapField="PER_ID"/`
			`>`
			`</persons>`

mapList=	"list name"	The **mapList** attribute is used only for business service mapping. It is used in two ways:
		• First, to create a list in the business service that corresponds to a list in the service.
		• Second, you can use **mapList** to identify the list that a flattened field lives in. For more information on flattening, refer to flattening section below.
			Example of a list within a business service:
			`<selectList type="list" -`
			`  mapList="DE" -`
			`  <value -`
			`mapField="COL_VALUE"> -`
			`  <row mapList="DE_VAL"> -`
			`  <SEQNO is="2"/> -`
			`  </row> -`
			`  </value>  -`
			`</selectList>`

mapXML=	"field name"	The **mapXML** attribute is typically used to map XML structures into a large character / XML field of the service. Note that when you use **mapXML** to map either a list or group structure (**type="list"** or **type="group"**) you don't have to map all the child elements within the structure. It is also possible to map list elements to a large character field associated with the list child.	`<enrollmentRequest -`
			`type="group" -`
			`mapXML="CASE_CLOB">`
			`  <messageID/>`
			`  <sender/>`
			`</enrollmentRequest>`
			`<enrollmentKey -`
			`mapXML="CASE_CLOB"/>`
			`<enrollmentInfo type="list" -`
			`mapChild="CI_CASE_CHILD">`
			`  <sequence -`
			`  mapField="CHILD_SEQ"/>`
			`  <name -`
			`  mapXML="CASE_CHILD_CLOB"/>`
			`  <value -`
			`  **mapXML="CASE_CHILD_CLOB"**/>`
			`</enrollmentInfo>`

isPrimeKey=	**"true"**	You must specify a primary key for a list defined within a mapped XML element (**type="list" mapXML="CLOB"**). The primary key is used by the framework to determine whether a list element add, update or delete	`<questionnaire -`
			`type="list" -`
			`mapXML="CASE_CLOB">`
			`  <question -`
			`  isPrimeKey="true"/>`
			`  <answer/>`
			`</questionnaire>`

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Valid Values	Description	Examples
		is required during a business object update. **Note:** You do not need to specify the prime key for a business object list mapped to maintenance object list. When a physical list is mapped, the prime key is derived from existing physical meta-data.	

orderBy=	"XPath **asc|desc**, XPath **asc|desc**"	By default, a list defined within a mapped XML element (**type="list" mapXML="CLOB"**) is sorted by the first element of the list. A different sort order may be specified using the **orderBy** attribute. The attribute value is a comma separated list of field XPaths (relative to the list element) with an optional sort order (ascending is the default).
		
		Note: This is only available for lists mapped as XML within business objects.
			`<questionnaire type="list" - orderBy="page/section, - page/sequence" - mapXML="CASE_CLOB">`
			`<question isPrimeKey="true"/>`
			`<answer/>`
			`<page type="group">`
			`  <section/>`
			`  <sequence/>`
			`</page>`
			`</questionnaire>`

Descriptive Attributes

The following attributes can be used to describe a schema element and provide additional configuration related to the element. Typically, these attributes are useful for field elements only.

Mnemonic	Valid Values	Description	Examples

<!-- comment -->		Use this to add a comment to a schema by using special open and close characters: <!-- and -->.	`<schema>`
			`  <!-- This schema is used to capture business information only, please refer to the 'HUMAN' BO for person information --->`
			`</schema>`

description=	"text"	The description of an element may be used to provide an internal description of the element to help a reader of the schema to understand the business reason for the element.	`<schema>`
			`  <active type="boolean" - description="active account" - label="Active"/>`
			`</schema>`

label=	"text"	The label of an element is meant to be a short bit of verbiage that would typically precede the element in a user interface.	`<schema>`
			`  <active type="boolean" - description="active account" -`
			`**label="Active"**/>`
			`</schema>`

required=	**"true"**	Used to require the existence of an element during object interaction.	`<schema>` `  <logDate mapField="LOG_DT" -`

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Valid Values	Description	Examples
		Note: For included schemas, the required="true" attribute is not processed on business object and business service schemas when they are included within a service script schema. This is to support the ability of the service script to populate required elements before an embedded business object or business service is invoked from the service script.	`default="%CurrentDate" - required="true" - private="true" </schema>`
mdField=	"field code"	The meta-data field attribute is used to associate an element with a field's metadata. The field defines data type, as well as its label and help text. If you link an element with a meta-data field, you don't need to specify any of these attributes. **Note:** For a business object schema, the mapField attribute is used to derive the data type and label. An mdField attribute may be provided to override these attributes, if needed. If the element is mapped to an XML column, the mdField is needed to provide the appropriate data type and label.	`<schema> <active - mdField="CM_ACTIVE_SW"/> </schema>`
fkRef=	"FK Reference Code"	If the element is a foreign key, defining its FK Reference will enable framework validation of the element during schema interaction and automatically provide descriptions and navigation capability.	`<schema> <person fkRef="PER" - mapField="CHAR_VAL_FK1"> <row mapChild="CI_SA_CHAR"> <CHAR_TYPE_CD is="PER"/> </row> </person> </schema>`
private=	**"true"**	Marking an element as private will prevent it from being exposed in schema interaction. **Note:** A private element requires a default.	`<schema> <type mdField="SA_TYPE_CD" - default="E1" private="true"/> </schema>`
suppress=	**"true"**	This setting prevents an element from appearing in automatically generated user interfaces.	`<schema> <ls - mdField="LIFE_SUPPORT_FLG" - default="N" suppress="true"/< </schema>`

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Valid Values	Description	Examples
		Note: This attribute can be specified on a group, in which case all elements of the group will be suppressed. **Note:** Refer to Defaulting and System Variables for advice related to default values for suppressed elements.	

	"blank"	This setting means that automatic UI rendering in display mode will hide the element if its value is blank. The element will still be modifiable on the input map whether blank or not.	`<schema>`
			`  <email mdField="EMAIL" -`
			`suppress="blank"/>`
			`</schema>`

	"input"	This setting means that automatic UI rendering will suppress the element for input, although it may still be displayed if it is not blank.
		
		Note: Elements marked as **suppress="input"** will behave as with **suppress="blank"**. If the value is blank, no value will be displayed on either the display or input map. If the element's value is present, then the element will be displayed on both the display and input map.
		
		Note: Refer to Defaulting and System Variables for advice related to default values for suppressed elements.
			`<schema>`
			`  <email mdField="EMAIL" -`
			`suppress="input"/>`
			`</schema>`

noAudit=	**"true"**	This setting prevents an element from appearing as a changed element in the business object's audit plug-in spot. If specified on a group or list node it applies to the whole node. You cannot specify it on the schema root node, only on schema elements.
		
		Note: This attribute is only applicable to business object schemas.
			`<schema>`
			`  -...`
			`  <version -`
			`mapField="VERSION_NBR"`
			` noAudit="true"/>`
			`  -...`
			`  <workFields type="group"`
			` noAudit="true"`
			`  <lastProcessedId/>`
			`  <lastProcessedTime/>`
			`  </workFields>`
			`</schema>`

storeEmptyNodes=	**"true"**	By default, empty nodes are removed from a business object instance when saved. This setting	`<schema>`
			`  <message type="group" -`
			`storeEmptyNodes="true">`
			`  -...`

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Valid Values	Description	Examples
		allows a group or a list element to explicitly keep its empty nodes. This attribute may be useful in situations where business object data is exchanged with an external system and there is a need to distinguish between an empty value for an element that participates in the synchronization and an element that does not participate and therefore omitted from the message altogether. **Note:** This is only available for group and list elements.	`</message>` `</schema>`

emitEmptyGroups=	"true"	By default, empty nodes are not included in the output of a business object or a business service call. This attribute is set at the top level schema node, allowing empty nodes of all groups and list elements to be included in the output.
		
		This attribute may be useful in the same situations that may require the use of the **storeEmptyNodes=** attribute.
		
		Note: This is only available for business object and business service schemas.
			`<schema -`
			`emitEmptyGroups="true">`
			`<message type="group">`
			`-...`
			`</message>`
			`</schema>`

emitEmptyElements=	"true"	This attribute is similar to the **emitEmptyGroups=** attribute but may also be specified at the field level. The attribute is applicable to all schema types.	`<schema>`
			`<messageNumber -`
			`type="field" -`
			`emitEmptyElements="true">`
			`-...`
			`</message>`
			`</schema>`

adheresToDA=	"comma separated list of data area names."	This attribute is applicable to the schema, group and list nodes. It claims that this sub-schema node is intended to "look like" the schema of one of the listed data areas, for polymorphism reasons in general.	`<schema>`
			`<messageNumber -`
			`type="group" -`
			`adheresToDA="DataAreaName">`
			`-...`
			`</message>`
			`</schema>`

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Valid Values	Description	Examples
		Note: The recommended approach for enforcing the structure of a shared data area is by including it in the schema. This attribute should be limited to use cases where the ability to include the data area is not possible.	

Schema Constants

There are some product owned schemas where the design warrants a value to be defaulted in the schema, but where the value is implementation specific and therefore cannot be defined by the product. For these situations, the product may use a technique called a schema constant. The design of the schema will include a declared constant. At implementation time, a configuration task will include defining the appropriate value for the constant.

For example, imagine the product delivers an algorithm that will create an outbound message when a certain condition occurs. The outbound message type to use must be configured by the implementation. To use a schema constant to define the outbound message type, the base product will configure the following:

- An option type lookup value for the lookup **F1CN_OPT_TYP_FLG** is defined. For example, **M202** - Activity Completion Outbound Message Type with a Java Value Name of **outmsgCompletion**
- The base schema that is used to create the "complete activity" outbound message references the schema constant using the Java Value Name of the option type's lookup value

```
...
<outboundMessageType mapField="OUTMSG_TYPE_CD" default="%Constant(outmsgCompletion)"/>
...
```

At implementation time, the administrative users must configure the appropriate outbound message type for "activity completion". Then, navigate to Feature Configuration, choose the **Schema Constants** feature type, choose the option type **Activity Completion Outbound Message Type** and enter the newly created outbound message type in the option value.

Schema constants may also be used in the flattening syntax to define the row elements required for flattening.

Defaulting and System Variables

The default attribute can be used to default values into field elements as well as the row elements required for flattening . You can default a field to a constant or to one of several system variables.

When an element is required, the default value is applied at the server level when adding or changing the record where no value is provided for this element.

When an element is optional, a default value may be configured to provide a suggested value to the user when populating the record on the user interface in Add mode. Users should be able to remove the default value for this type of element and the system will not attempt to populate it at save time.

Copyright © 2024, Oracle and/or its affiliates.

Note:

The standard for our 'list' maintenance is to show an empty row for a new list that a user may or may not choose to populate. As such, UI default values are not shown for list elements.

It is not recommended to configure a default attribute for an optional element that is not editable or visible on the user interface in Add mode (for example an element with **suppress="true"** or **suppress="input"** or a list element whose default value is not shown per the previous point). The default for this type of element is still applied but the user doesn't have the ability to reset the value. This may result in inconsistencies as defaulting does not occur for optional fields added by a use case other than the UI interaction.

Mnemonic	Valid Values	Description	Examples

default=	"value"	Use this attribute to default an element to a specified value. The values that are valid depend on the **dataType** setting.	`<schema>`
			`  <perType -`
			`mapField="PER_OR_BUS_FLG" -`
			`default="P" -`
			`required="true"/>`
			`</schema>`

			`<schema>`
			`  <frequency -`
			`dataType="number" -`
			`default="1" -`
			`required="true"/>`
			`</schema>`

	"%CurrentDate"	Used to default the element to the current date. This is only applicable to date elements.	`<schema>`
			`  <logDate -`
			`mapField="LOG_DT" -`
			`default="%CurrentDate" -`
			`required="true"/>`
			`</schema>`

	"%CurrentDateTime"	Used to default the element to the current date / time. This is only applicable to date / time elements.	`<schema>`
			`  <logDateTime -`
			`mapField="LOG_DTTM" -`
			`default="%CurrentDateTime" -`
			`required="true"/>`
			`</schema>`

	"%StandardDateTime"	Used to default the standard date and time. The standard date and time is identical to the current date and time, unless daylight savings time / summer time is in effect for the base time zone. This may be used with the **stdTime** attribute.
		
		Note: Refer to Standard Time Considerations for more information.
			`<schema>`
			`  <startDateTime -`
			`mapField="START_DTTM" -`
			`default="%StandardDateTime" -`
			`required="true"/>`
			`</schema>`

	"%ProcessDate"	You can default the process date. The process date differs from the current date because the process date will remain constant throughout the duration of the process	`<schema>`
			`  <billDate -`
			`mapField="BILL_DT" -`
			`default="%ProcessDate" -`
			`required="true"/>`
			`</schema>`

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Valid Values	Description	Examples
		being executed. The current date will reflect the actual date of processing. This is similar to the batch business date that is a standard batch parameter.	

	"%ProcessDateTime"	This is similar to **"%ProcessDate"** but for date / time fields.	`<schema>`
			`  <calcDateTime -`
			`mapField="CALC_DTTM" -`
			`default="%ProcessDateTime" -`
			`required="true"/>`
			`</schema>`

	"%CurrentUser"	Used to default the element to the current user.	`<schema>`
			`  <logUser -`
			`mapField="LOG_USER" -`
			`default="%CurrentUser" -`
			`required="true"/>`
			`</schema>`

	"%CurrentUserTimeZone"	Used to default the element to the current user's time zone. If the current user's time zone is not found, the installation time zone is used.	`<schema>`
			`  <timeZone -`
			`default="%CurrentUserTimeZone" -`
			`required="true"/>`
			`</schema>`

	"%CurrentUserLanguage"	Used to default the element to the current user's language.	`<schema>`
			`  <custLanguage -`
			`mapField="CUST_LANG" -`
			`default="%CurrentUserLanguage" -`
			`required="true"/>`
			`</schema>`

	"%InstallationCurrency"	Used to default the currency from the installation record.	`<schema>`
			`  <currency -`
			`mapField="CURRENCY_CODE" -`
			`default="%InstallationCurrency"` `-`
			`required="true"/>`
			`</schema>`

	"%InstallationCountry"	You can default the country from installation record.	`<schema>`
			`  <country -`
			`mapField="COUNTRY" -`
			`default="%InstallationCountry" -`
			`required="true"/>`
			`</schema>`

	"%InstallationLanguage"	You can default the language from the installation record.	`<schema>`
			`  <language -`
			`mapField="LANGUAGE" -`
			`default="%InstallationLanguage"` `-`
			`required="true"/>`
			`</schema>`

	"%Constant( )"	You can default an element value using a schema constant.	The following is an example of a schema constant used as a default value, where the Java Value

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Valid Values	Description	Examples
			Name of the Lookup Value is 'customerLanguage'. `<language - mapField="CUSTOMER_LANG" - default="%Constant(customerLangu required="true"/>`
	"%Context( )"	You can default a value contained in a context variable. **Warning:** Context variables must be initialized within a server script before the schema context default can be applied. Refer to Context Variables for more information.	An example of a context variable used as a default value: `<source - mapField="PER_ID" - **default="%Context(source)"** - required="true"/>` **Note:** When defining a context variable in scripting, it should be prefixed with **$$**. When referring to the variable in the **%Context( )** syntax, the prefix is not included.
defaultRef=	"XPath"	Use this attribute to default the value of one element to the value of another one.	Refer to Referencing Other Elements for supported syntax for referring to other elements.

The Flattening Nodes and Attributes

The term "flattening" is used to describe the act of defining one or more single elements for a schema that are actually part of a list within the maintenance object. Flattening is possible if there are other attributes of the list that can be defined to uniquely describe the element or elements. A common use case for flattening is a characteristic. Rather than defining the characteristics of an object using a collection where the user must choose the characteristic type and then define the value, the characteristics are defined as actual elements with the appropriate label already displayed. This technique enables the designer of the schema and the user interface to display each separate characteristic in the logical place in the user interface rather than all lumped together.

Note: A flattened element represents a unique row in the database. This row is inserted when the flattened values are created. The row is updated when any of the flattened values are changed. The row is deleted when all the flattened values are removed. The behavior of effective dated elements is slightly different - please see Flattening an Effective Dated List.

Note: The flattening feature can also be used to define a list, see Flattened List.

Identifying the List or Child Table

When flattening a child table, the row node is required to identify the list / child table that the element comes from. Within the row node, at least one element must be defined with an **is=** definition that ensures that the element is a unique row in the database. It may also define elements or fields in the row that are suppressed and are populated using default value configuration.

- For a business object, the row node defines the child table the flattened field belongs to.

  The syntax is **<row mapChild="table name">**. This example is for the list of persons for an account in the customer care and billing product. One person may be marked as the "main" person. This illustrates how to define an explicit

Copyright © 2024, Oracle and/or its affiliates.

element for the main person ID to simplify references to that field. It is part of the CI_ACCT_PER child table. What makes it unique is that the MAIN_CUST_SW is **true** (and only one row may have that value)

```
<custId mapField="PER_ID" mdField="CM-MainCust">
  <row mapChild="CI_ACCT_PER">
    <MAIN_CUST_SW is="true" -/>
    <ACCT_REL_TYPE_CD default="MAIN" -/>
  </row>
</custId>
```

Note: The above example illustrates that the row node may also define elements within the list that are suppressed and assigned a default value. This syntax is never used to identify a particular row. Note that a default value can either be a literal string, or a system variable.

- For a business service, the row node identifies the list name the flattened field belongs to.

  The syntax is **<row mapList="list name">**. This example shows two entries from a list being flattened to a field value and description.

```
<selectList type="list" mapList="DE"> -
    <value mapField="COL_VALUE"> -
 <row mapList="DE_VAL"> -
          <SEQNO is="2"/> -
 </row> -
    </value>  -
  <description mapField="COL_VALUE"> -
 <row mapList="DE_VAL"> -
        <SEQNO is="3"/> -
 </row> -
  </description> -
</selectList>
```

Uniquely Identifying the Flattened Field or List

The **is=** syntax within a row or rowFilter element is used to uniquely identify the row.

Mnemonic	Valid Values	Description	Examples

is=	"value"	Use this attribute to reference a value directly.	`<tdTypeCd -`
			`mapField="CHAR_VAL_FK1">`
			`<row -`
			`-`
			`mapChild="F1_EXT_LOOKUP_VAL_CHAR">`
			`<CHAR_TYPE_CD -`
			`**is="CM-TD-TYPE"**/>`
			`</row>`
			`</tdTypeCd>`

	"%Constant( )"	You can configure a flattened element using a schema constant. During a service interaction the value of the schema constant will be used to identify the flattened element in its child row.	An example of a schema constant used in flattening syntax where the Java Value Name of the Lookup field value is 'cmRate'.
			`<unitRate mapField="CHAR_VAL" -`
			`dataType="number">`
			`<row mapChild=`
			`-"F1_EXT_LOOKUP_VAL_CHAR">`
			`<CHAR_TYPE is`
			`**="%Constant(cmRate)"**/>`
			`</row>`
			`</unitRate>`

	"%n"	The %n substitution value is used to reference a relative list instance. A relative list instance is typically used to configure a flattened element for a child table	An example with a relative list instance - where the **first** instance of the row is returned.
			`<schema>`
			`<mainPhone mapField="PHONE">`
			`<row mapChild="CI_PER_PHONE">`

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Valid Values	Description	Examples
		keyed by sequence number. The value of *n* should be a positive integer value. During a business object read interaction the relative list instance (position) specified by the integer will be returned.	`<PHONE_TYPE_CD - -` `is="%Constant(mainPhoneType)"/>` `<SEQ_NUMis="%1"/>` `</row>` `</mainPhone>` `</schema>`

Fastpath: Additional values for **is=** are used when Flattening an Effective Dated List. Refer to that section for more information.

Flattening a Pre-defined Characteristic Type

If the flattened field is in a characteristic collection and the characteristic type is a predefined characteristic, automatic UI rendering will generate a dropdown for the list of valid values. For example, the schema below will generate a dropdown for the Usage element showing the valid values of the Status Reason Usage (**F1-SRUSG**) characteristic type.

```
<usage mdField="STATUS_RSN_USAGE" mapField="CHAR_VAL">
  <row mapChild="F1_BUS_OBJ_STATUS_RSN_CHAR">
      <CHAR_TYPE_CD is="F1-SRUSG"/>
      <SEQ_NUM is="1"/>
  </row>
</usage>
```

Defining Multiple Elements from the List

When attempting to include multiple columns from the same list, the system provide shorthand notation for copying the flattening rules defined on another element so that the flattening rules do not need to be repeated. To do this, the row node includes the **rowRef** attribute and it indicates the other element name that defines the mapping information. The following example illustrates flattening the fields Customer ID and Receives Copy of Bill from the same list of Persons for an Account (where the MAIN_CUST_SW is **true**).

```
<custId mapField="PER_ID">
  <row mapChild="CI_ACCT_PER">
      <MAIN_CUST_SW is="true" -/>
      <ACCT_REL_TYPE_CD default="MAIN" -/>
  </row>
</custId>
<copyBill mapField="RECEIVE_COPY_SW" rowRef="custId"/>
```

Note that the above notation also illustrates that the **rowRef** may be defined directly in the element's attribute definition.

Note: Refer to Referencing Other Elements for supported syntax for referring to other elements.

Flattening Two Layers Deep

If your maintenance object or service has nested lists two layers deep, the system supports flattening an element within a flattened element. This technique also uses the **rowRef** attribute. The flattening of the second level refers to the flattened element of the first level. The following example illustrates flattening a characteristic into an element called legalContact for the "main" customer. Notice that the legalContact element has two row nodes: one to refer to the flattening information for its parent record and one to define its child table

```
<custId mapField="PER_ID">
  <row mapChild="CI_ACCT_PER">
    <MAIN_CUST_SW is="true" -/>
    <ACCT_REL_TYPE_CD default="MAIN" -/>
  </row>
</custId>
<legalContact mapField="CHAR_VAL_FK1">
  <row rowRef="custId">
    <row mapChild="CI_ACCT_PER_CHAR" >
       <CHAR_TYPE_CD is="LEGAL" -/>
    </row>
```

Copyright © 2024, Oracle and/or its affiliates.

```
  </row>
</legalContact>
```

Note that the above notation also illustrates that the **rowRef** may be defined as an attribute of a row node rather than directly in the element's attribute definition.

Defining a Flattened List

There are times that a list or child table supports multiple values of the same "type" and these should be presented as a list. To continue with the example above, the list of persons for an account may identify one person as the "main" person. This person has been flattened to a single element (with the account relationship type defaulted and suppressed). To maintain the other persons related to an account, you can define a list where each row captures the Person Id and the Account Relationship Type.

Rather than a row node, the flattened list is configured with a **rowFilter** element. The following schema illustrates the described example. The list node defines the child table. The **rowFilter** includes the criteria that identify the rows within the table to include. The elements of the list are defined within the list node outside the **rowFilter** element.

```
<custId mapField="PER_ID">
  <row mapChild="CI_ACCT_PER">
    <MAIN_CUST_SW is="true" -/>
    <ACCT_REL_TYPE_CD default="MAIN" -/>
  </row>
</custId>
<miscPersons type="list" mapChild="CI_ACCT_PER">
  <rowFilter>
    <MAIN_CUST_SW is="false" -/>
  </rowFilter>
  <relType mapField="ACCT_REL_TYPE_CD"/>
  <personId mapField="PER_ID"/>
</custId>
```

Note that the system will validate that if a schema contains flattened single elements and flattened lists from the same child table, the criteria that defines what makes them unique must be analogous.

Flattening an Effective Dated List

There are some lists in the application that are effective dated (and still others that have effective date and time). For example, there are some effective dated characteristic collections. In these collections, the design is to capture a single value for a characteristic type that may change over time. It is not meant to support multiple characteristic values in effect at the same time.

There are some maintenance objects that have effective dated characteristics and no sequence based characteristics. For those entities, implementations may want to use the characteristic to define an additional field that doesn't require effective dates.

When considering designing an element for your schema that represents a value in an effective dated collection, consider if the value is one that is used in processes, like calculating a bill, such that it is important to capture changes to the value over time.

- If it is not important to track the changes to the value, the recommendation is to set the effective date to a fixed value in the schema. (Examples below).
- If it is important to track the changes to the value, then you need to consider how you want to design the schema.
  - One option is to define a flattened list as described above. Both the value and the effective date are elements in this list. In this use case, you manage the list just like any list. You can add or remove values.
  - Another option is to define a flattened single value in the schema. With this option, when retrieving the record, the latest effective value is returned. In this design, the important question is how is the reference date defined.

Note: Refer to Referencing Other Elements for supported syntax for referring to other elements.

Flattened Elements Where Changes Are Not Tracked

Copyright © 2024, Oracle and/or its affiliates.

The following is an example of flattening an element in an effective dated collection where the value does not need to be effective dated. The recommendation in this case is to simply hard-code the date in the **is=** attribute.

```
<schema>
  <externalReference mapField="ADHOC_CHAR_VAL" mdField="CM_EXT_REF" -
dataType="string">
    <row mapChild="CI_SA_CHAR">
      <CHAR_TYPE is="CM_EXT_REF"/>
      <EFFDT is="2000-01-01"/>
    </row>
  </externalReference>
</schema>
```

Flattened Elements Where Changes Are Effective Dated

If you want the BO schema to flatten an effective dated value that should keep track of changes over time, the flattened value will represent the latest effective entry. In this case, the recommendation is to explicitly define the date or date/time field as an element. Then there is syntax for the **is=** attribute to indicate that the explicit date field is the date to use for the value.

Mnemonic	Valid Values	Description	Examples

is=	**"%effectiveDate( )"**	Use this configuration to indicate that the date to use is the value of another element.
		Note: Refer to Referencing Other Elements for supported syntax for referring to other elements.
			`<schema>`
			`  <price -`
			`mapField="CHAR_VAL" -`
			`  dataType="number" -`
			`required="true">`
			`  <row -`
			`mapChild="CI_SA_CHAR">`
			`  <CHAR_TYPE -`
			`is="PRICE"/>`
			`  <EFFDT`
			`  -`
			`is="%effectiveDate(priceEdate)"/`
			`>`
			`  </row>`
			`  </price>`
			`<priceEdate -`
			`mapField="EFFDT"`
			`  rowRef="price"/>`
			`</schema>`

	"%effectiveDateTime( )"	Use this configuration to indicate that the date / time to use is the value of another element.
		Note: Refer to Referencing Other Elements for supported syntax for referring to other elements.
			`<schema>`
			`  <price -`
			`mapField="CHAR_VAL" -`
			`  dataType="number">`
			`  <row -`
			`mapChild="RATE_CHAR">`
			`  <CHAR_TYPE -`
			`is="PRICE"/>`
			`  <EFFDTTM is=`
			`  -`
			` -"%effectiveDateTime`
			`  (priceEdatetime)"/`
			`>`
			`  </row>`
			`  </price>`
			`  <priceEdatetime -`
			`mapField="EFFDTTM"`
			`  rowRef="price"/>`
			`</schema>`

You should consider if the explicit date or date/time element should be displayed on the user interface when the flattened field is shown or if that should be suppressed. Designers should also consider the user dialogue for updating the value. It may be more intuitive for the flattened field to be display only and to have a separate dialogue for updating the effective dated value.

Copyright © 2024, Oracle and/or its affiliates.

For this option, the suggestion is that the maintenance page for the object includes a separate zone that shows the history of the effective dated values over time. The latest value may be included in that zone as well.

Note that for backward compatibility, the system also supports the following options. However, they are not recommended. They implicitly defined the reference date as the current date at all times. The options supported adding new effective dated rows when the value changed. But the technique also caused new effective dated rows to appear when other elements of the same schema were updated. The new rows would have the current date or date/time with the same row value because the reference date or date/time is the current date, which is a change. The product does not recommend managing an effective dated characteristic without defining the date or date/time as an explicit element.

Mnemonic	Valid Values	Description	Examples

is=	**"%effectiveDate"**	This configuration is not recommended	`<schema>`
			`  <price mapField="CHAR_VAL" -`
			`  dataType="number">`
			`  <row -`
			`  mapChild="CI_SA_CHAR">`
			`  <CHAR_TYPE is="PRICE"/>`
			`  <EFFDT`
			`  is="%effectiveDate"/>`
			`  </row>`
			`  </price>`
			`</schema>`

	"%effectiveDateTime"	This configuration is not recommended	`<schema>`
			`  <price mapField="CHAR_VAL" -`
			`  dataType="number">`
			`  <row mapChild="RATE_CHAR">`
			`  <CHAR_TYPE is="PRICE"/>`
			`  <EFFDT -`
			`  is="%effectiveDateTime" -/`
			`>`
			`  </row>`
			`  </price>`
			`</schema>`

Search Zone

A UI Map schema element can be configured to enable an automatic search dialog when the schema is included within a maintenance UI map.

Note: Please note that an fkRef can be configured with a search zone. If a schema element has an fkRef but no explicit search attributes (as described here) then the fkRef search information will be used in the UI map. In other words, if the schema element already has an fkRef, then these explicit search attributes in the schema are only used to override the fkRef search information.

Note: Refer to the UI Map Attributes and Functions for more information on search zone configuration.

search="search zone"

The search attribute can be used within a UI map schema and is used to automatically generate the oraSearch UI map attribute. The search zone is an explorer zone configured as a search.

```
<person fkRef="PER" search="C1_PSRCH"/>
```

searchField="search field:element|'literal';"

The searchField attribute can only be used in conjunction with the search attribute. The searchField attribute is used to build the oraSearchField UI map attribute. The searchField value is used to populate a search zone filter with an initial value when the search zone is launched. The initial value can be a literal also. The searchField value is used to match to the filter mnemonic also named searchField.

Copyright © 2024, Oracle and/or its affiliates.

Search field: element|'literal'. The search field represents the search zone filter to populate on launch. The element is the map's schema element used to populate the filter. The element is optional, if left blank, it will default to the element that this attribute is specified on. The searchField also takes 'literal' as input value

Note: Multiple filters can be populated within the search zone at launch, but multiple search field pairs must be constructed within the attribute value. The value specified here will be used to directly build the HTML attribute oraSearchField within the UI map where this schema is specified.

Note: Note that the element reference is *relative*. Refer to Referencing Other Elements for supported syntax for referring to other elements.

```
<person fkRef="PER" search="C1_PSRCH" searchField="PERSON; PER_TYPE:personType;"/>
```

searchOut="search field:element;"

The searchOut attribute can only be used in conjunction with the search attribute. The searchOut attribute is used to build the oraSearchOut UI map attribute. The searchOut value is used to capture a selected value from the search zone and move it to a UI map element. The searchOut value specified should match the ELEMENT_NAME mnemonic within the search result zone parameter.

Search field: element. The search field represents the search zone result brought back to the UI map. The element is the map's schema element to be populated. The element is optional, if left blank, it will default to the element that this attribute is specified on.

Note: Multiple elements can be populated as a result of search zone selection, but multiple search field pairs must be constructed within the attribute value. The value specified here will be used to directly build the HTML attribute oraSearchOut within the UI map where this schema is specified.

Note: Note that the element reference is *relative*. Refer to Referencing Other Elements for supported syntax for referring to other elements.

```
<person fkRef="PER" search="C1_PSRCH" searchField="PER_ID" -
searchOut="PER_ID;PRIMARY_PHONE:personPhone;"/>
```

Extend Security for Service Script

Application service security will be enforced when either a business object or a service script is invoked from a BPA script or a UI map, but not from a service script. If you want security to be enforced when the business object or a service script is invoked from a service script, you must add the following attribute to the service script's schema.

appSecurity="true"

The appSecurity attribute is only available for service script schemas. If specified, any business object or service script directly invoked by the service script will have their application service evaluated for access.

```
<schema appSecurity="true">
  -...
</schema>
```

Overriding Action for a Business Service

If you want to invoke a business service with an action other than 'read', you need to specify the action attribute on the primary node business service schema.

pageAction="add, change, delete"

The action attribute is used to override the default action of read on a business service schema. Valid values are:

- add
- update (only allowed for maintenance object service)

Copyright © 2024, Oracle and/or its affiliates.

- change (not allowed for maintenance object service)
- delete
- read (this is the default action if no pageAction specified)

Example:

```
<schema pageAction="change">
  <parm type="group">
    <ele1/>
    <ele2/>
  </parm>
</schema>
```

Specifying searchBy for a Search Service

If you want to invoke a search service then you must explicitly specify the searchBy attribute appropriate for the elements mapped in the schema.

searchBy="MAIN"

The value values of the searchBy attribute can be found by viewing the XML schema linked to the business service, use the View XML URL. Typical values are:

- MAIN
- ALT
- ALT2
- ALT3
- etc.

```
<schema searchBy="MAIN">
  <AccountID mapField="ACCT_ID"/>
  <Results type="list">
    <AccountID mapField="ACCT_ID"/>
  </Results>
</schema>
```

Including Other Schemas

There are no limitations on your ability to include a schema into another schema - all types can be included in all other types. Nested includes are also allowed - and at present there is no limitation on the depth of the nesting.

Including a schema requires two parts:

1. The include node
2. The name attribute

The following table highlights the supported include statements.

Mnemonic	Description	Examples

<includeBO name=" "/>	Including a business object schema into another schema is allowed.
	
	Note that the mapping rules of a business object or business service schema may or may not make sense in the context of the parent schema. Include other schemas at your own risk. However, a very useful aspect of XML processing is that the framework ignores non-pertinent attributes. In other words, it will
		`<schema>`
		`  <cust type="group">`
		`  <includeBO name="C1-Person"/>`
		`  </cust>`
		`  <spouse type="group">`
		`  <includeBO name="C1-Person"/>`
		`  </spouse>`
		`</schema>`

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Description	Examples
	not hurt to have mapping attributes included into a script schema.	

<includeBS name=" "/>	Used to include a business service schema.	`<schema>`
		`  <includeBS name="F1-ReadMOLog"/>`
		`</schema>`

<includeDA name=" "/>	Used to include a data area schema.	`<schema>`
		`  <includeDA -`
		`  name="F1CommonSchemaFieldData"/>`
		`</schema>`

<includeMP name=" "/>	Used to include a UI map schema.	`<schema>`
		` <includeMP -`
		`  name="F1-DisplayRecordActions"/>`
		`</schema>`

<includeSS name=" "/>	Used to include a service script schema.	`<schema>`
		` <includeSS name="F1-ActShowZn"/>`
		`</schema>`

Compatibility Attributes

These attributes were added as part of upgrades from previous versions of the Framework.

fwRel="2"

This attribute has been added to schemas created in Framework 2 as part of a Framework 4 upgrade. New schemas will not need this attribute. It is not advisable to modify this attribute without understanding the following behavior differences as improper changes could result in errors:

Note: Schemas created in Framework 2 with the fwRel="2" attribute will store any XML mapped fields under groups as top-level XML elements in the mapXML field. This means that if two or more fields, in different group structures, were to have the same field name, their storage would conflict with one another resulting in errors. The new behavior, without the fwRel="2" attribute, will preserve the group structure and avoid the conflicts.

```
<schema fwRel="2"
    -...
</schema>
```

UI Hint Syntax

Working Examples

For working examples of uiHint functionality, refer to the following business objects:

BOs with User Assigned Keys

The following examples illustrate the patterns used to enable uiHints on an object with a user specified key.

- **F1-OutcomeStyleLookup**. This extendable lookup BO does not require state transition, but does allow duplicate and delete actions.
- **F1-TodoSumEmailTyp**. This request type illustrates the hints required to support state transition on a display map.
- **F1-WebSvc**. This web service BO is a good example for management of complex JavaScript requirements. Both display and input maps have functionality that requires specialized javascript.

Copyright © 2024, Oracle and/or its affiliates.

BO with System Generated Key

The following example illustrates the pattern used to enable uiHints on an object with a system generated key.

- **F1-GenericAttachment**. This attachment BO has a system assigned key, which entails the following special handling:
  - **F1-AttachmentMain**. This is the main section data area contains the elements common to all attachments, including the key, bo, and version. Because this data area is used to define the main section of the generated maps, the main section of the map can be extended by an implementation via data area extension functionality.
  - **F1-AttachmentActions**. This record actions map contains the standard actions, Edit and Delete, plus custom actions used only by attachments, View and Upload.
  - **F1-AttachmentIDFrag**. This record information map contains the primary key of the attachment.

Display Map Service Script

Display map service scripts can be fully supported via dynamic HTML generation. However, to help eliminate the need for a display service script, self-contained uiHint functionality has been developed to write the business object status and determine valid state transitions. So the two most common reasons to craft a display service script have been eliminated.

A typical reason to use a display pre-script is if you have an embedded map fragment that contains a business service schema. The display service script can be used to invoke the business service. Both the map fragment and the display service script must declare the business service schema to support this scenario.

Warning: The zone used to display the object's map must have a derivation script, like **F1-GncDsMpDZ** or **F1-GenDss**, that will invoke a display service script for the business object if it has been defined as a BO option - but not require an explicit display map BO option. In addition, the display service script's schema must be enabled for uiHint functionality - as the script's schema will be dynamically rendered by the zone - and not the BO schema.

- **F1-ExcelSpreadsheet**. This attachment BO has a display service script used to manipulate the attachment business object before displaying it:
- **F1-AttchDtlU**. This display map service script's schema has been defined with the uiHint namespace, and will have a display map generated for it.

Maintenance Pre-Processing Service Script

Maintenance pre-processing service scripts can be used with uiHints.

- **F1-ExcelSpreadsheet**. This attachment BO has a maintenance pre-processing service script used to manipulate the attachment business object before rendering the maintenance map:
- **F1-AttchPre**. This pre-processing service script's schema mimics a maintenance map schema with embedded boGroup and action elements. It will be invoked before the maintenance map is rendered.

Maintenance Post-Processing Service Script

Maintenance post-processing service scripts can be used with uiHints.

- **F1-ExcelSpreadsheet**. This attachment BO has a maintenance post-processing service script used to manipulate the attachment business object after rendering the maintenance map:
- **F1-AttchPost**. This post-processing service script's schema mimics a maintenance map schema with embedded boGroup and action elements. It will be invoked after the maintenance map is rendered.

Technical Notes

The following prerequisites are required to support dynamic HTML generation:

Schema Requirements

To support automated UI generation, the business object schema must contain the following:

- <schema **xmlns:uiHint="http://oracle.com/ouafUIHints"**>. The schema node must name the uiHint namespace.
- **isPrimeKey="true".** Every element of the business object schema that is part of the primary key must be identified.

Maintenance Script Requirements

Copyright © 2024, Oracle and/or its affiliates.

The maintenance script for the MO must be enabled for dynamic generation.

CAUTION: The business object maintenance BPA script must be declared as an MO Option for uiHint maintenance functionality to work!

If the script performs **F1-BOProc** then it is likely no special functionality is needed. However, if the maintenance script contains its own call to **F1-GetValOpt** then the following statement is required prior to that call:

```
move -'false' to -"F1-GetBOOpts/input/maintenanceMapRequired";
performScript -'F1-GetValOpt';
```

After the call to **F1-GetValOpt** the following logic must be included to dynamically declare the map schema if the business object does not have a maintenance map of its own:

```
 -// Perform Main Processing
if ("F1-GetBOOpts/output/maintenanceMap = $BLANK")
  declareBOWithBOGroup -"$bo" as -'map_schema';
else
  declareMap -"F1-GetBOOpts/output/maintenanceMap" as -'map_schema';
end-if;
```

Format an Input Map Title

Note: Throughout this topic the term "field" to refer to both the generic concept of displaying and capturing data in a 'field' as well as referring to the meta-data object supplied in the product to define Fields. When referring to the latter, the term "MD Field" (meta-data Field) is used.

A uiHint element can be used to build a title for a maintenance map. The title will only print on the maintenance map, not on the display map. It will be printed as the first line in the map, centered, with a heading style.

Syntax	Description	Examples

<uiHint:title mdField=" "/>	Displays the label of a referenced MD field as the title.	`<schema xmlns:uiHint="http://oracle.com/ouafUIHints">`
		`  <uiHint:title -`
		`  mdField="STATUS_RSN_LBL"/>`
		`  -...`
		`</schema>`

<uiHint:title text=" "/>	Displays the indicated text as the title. (Do not use this mechanism when multiple languages are supported.	`<schema xmlns:uiHint="http://oracle.com/ouafUIHints">`
		`  <uiHint:title -`
		`  text="Status Reason"/>`
		`  -...`
		`</schema>`

Create a Section

The uiHint namespace supports the definition of a UI map section. Note that sections are currently created in generated UI Maps when the schema has a group or list node with a label or mdField. The functionality described here enables the creation of a section without requiring a labeled group or list node within the schema. Every section must be bounded by **startSection** and **endSection** element pair.

Syntax	Supporting Attributes	Description
<uiHint:startSection/>	sectionColumn="left | right | fullWidth | float"	The default is that the section will be the full width in display maps. To override that setting, specify if you want a half-width section to appear in either the left (**left**) or right (**right**) column

Copyright © 2024, Oracle and/or its affiliates.

Syntax	Supporting Attributes	Description
		or to float (**float**). Sections that are marked as 'float' will display half-width and be aligned according to whether prior sections are displayed or conditionally hidden. For example, if a left-aligned section is followed by a floating section, the floating section will appear in the right column if the left section is populated but will display in the left column if the left section is hidden / collapsed.
	editColumn="left | right | fullWidth | float"	By default a section appears as full width in maintenance maps. To override that setting, specify if you want a half-width section to appear in either the left (**left**) or right (**right**) column or to float (**float**). The behavior is the analogous to the **sectionColumn** behavior.
	sectionOpen="false"	By default a section is open on initial display. Specify this attribute to initially display the section as closed (collapsed).
	mdField=" "	Specify the name of a MD field whose label should be used as the section heading.
	label=" "	Specify the explicit text to use as the section heading.
	visibleOn="displayMap | inputMap"	By default a section appears on both the display and the input maps. Use this attribute to limit the display of the section to either the display map (**displayMap**) or input map (**inputMap**).

The syntax for the end section attribute is **<uiHint:endSection/>**

Examples:

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
    <uiHint:startSection label="Main" sectionColumn="left"/>
    -...
    <uiHint:endSection/>
</schema>
```

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
    <uiHint:startSection mdField="F1-ADD-INFO" sectionColumn="fullWidth" -
editColumn="float" sectionOpen="false" visibleOn="displayMap"/>
    -...
    <uiHint:endSection/>
</schema>
```

Note: The sectionColumn, editColumn and sectionOpen attributes are available for group and list nodes as well.

Include a Map Fragment

You can specify a UI map fragment to inject HTML into a generated map using the **includeMap** element name. This allows for you to support more sophisticated behavior on your user interface. For any element that is included for rendering in the map fragment, be sure to suppress the element in its schema definition, otherwise HTML will automatically be generated for the element.

Copyright © 2024, Oracle and/or its affiliates.

Syntax	Supporting Attributes	Description
<uiHint:includeMap/>	map=" "	Specify the name of the map.
	visibleOn="displayMap | inputMap"	By default the details from the map fragment appear on both the display and the input maps. Use this attribute to limit the display of the section.

Example:

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
  -...
  <uiHint:includeMap map="StandardActionButtons" visibleOn="displayMap"/>
  -...
</schema>
```

Note: Important note on the map fragment schema: If a map fragment contains a schema, then the fragment schema structure will be injected into the dynamically generated schema when the business object is rendered for input. Technically, the fragment schema will be inserted after the boGroup structure within the map's schema. This method may be used to support the implementation of maintenance pre and post script processing for a business object and oraInvokeBS function calls within embedded JavaScript.

If JavaScript is required within an XHTML UI Map fragment, it is necessary to bound it within a **![CDATA[ ]]** tag to ensure a valid XML document. Note that the tags themselves may need to be commented out to promote compatibility with older browsers. For example:

```
  <script type="text/javascript">
  -/* <![CDATA[ */
  -//
  -//javascript
  -//
  -/* -]]> */
  </script>
```

Flush the cache: For performance reasons, the Framework automatically caches business object schemas, data areas, and UI maps. When you update a business object, the cache is automatically flushed. However, if the business object includes either a data area or embedded UI map fragment, the cache must be manually flushed in order for your changes to be recognized. Refer to Server Cache for more information.

Build A Dropdown

Syntax is provided to build a dropdown list in an edit map. The dropdown may be built using data returned from a service script, a business service or a table.

Syntax	Description
uiHint:select="ss: "	Specify the name of the service script after the colon.
uiHint:select="bs: "	Specify the name of the business service after the colon.
uiHint:select="table: "	Specify the name of the table after the colon.

When specifying a service script or a business service, extra mapping information is needed to pass data to and from the service.

Copyright © 2024, Oracle and/or its affiliates.

Syntax	Values	Description
uiHint:selectIn=" "	serviceXPath:element	Used to pass the value of another element into the service (mapping to the service's XPath).
	serviceXPath:'Literal'	Used to pass a constant or literal to the service (mapping to the service's XPath).
uiHint:selectOut="valuePath: ; descPath: "	See examples below.	Used to indicate which element in the service's output holds the values and which one holds the descriptions.

Examples:

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
    <boStatus mapField="BO_STATUS_CD" uiHint:select="bs:F1-BOStateReasonList" -
    uiHint:selectIn="boStatusBO:boStatusBO" uiHint:selectOut="valuePath:results/
status; -
    descPath:results/description"/>
    -...
    <algorithm mdField="ALG_CD" uiHint:select="bs:F1-RetrieveSysEvtAlgorithms" -
     uiHint:selectIn="algorithmEntity:'F1AA';" uiHint:selectOut="valuePath:results/
algorithm; -
     descPath:results/description"/>
    -...
    <outboundMsgType mdField="OUTMSG_TYPE_CD" required="true" fkRef="F1-OMTYP" -
uiHint:select="table:F1_OUTMSG_TYPE"/>
</schema>
```

Conditionally Hide Elements

The **displayNone** attribute is used to suppress elements on the map based on conditions.

Syntax	Values	Description
uiHint:displayNone=	"'XPath','value','!=' | '='"	Used to conditionally hide this element based on the value of another element (referenced using its XPath). Enter a value of ' ' to interrogate a blank value. By default the operator is **'='**. This may be overridden using **'!='**.
	"function name, **true** | **false**"	Used to indicate a JavaScript function, which must return a Boolean.

Warning:

Embedded spaces are not supported within the comma separated string values of this attribute.

This setting may be used on group nodes, list nodes, and elements - except for elements within a list. Elements within a list cannot be hidden conditionally.

The following example illustrates that two elements (currency reference and lookup) that will be hidden or displayed based on the value of the data type element. Note that this example also illustrates Trigger Dependent Behavior because the data type element's value may change and if it does, the condition for hiding the subsequent elements should be re-evaluated.

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
    -...
    <dataType mdField="F1_SE_DATA_TYPE" dataType="lookup" lookup="F1_SE_DATA_TYPE" -
     uiHint:dependents="currencyRef;lookup; -"/>
     <currencyRef mdField="F1_SE_CURR_REF_LBL" -
uiHint:displayNone="'dataType','F1MO','!='"/>
```

Copyright © 2024, Oracle and/or its affiliates.

```
     <lookup mdField="F1_SE_LOOKUP_LBL" fkRef="F1-LKUPF" -
uiHint:displayNone="'dataType','F1LP','!='"/>
     -...
</schema>
```

The following example illustrates referring to a function where the function receives parameters:

```
<uiHint:startSection mdField="F1_SE_DEFAULT_SECT" -
    uiHint:displayNone="isApplicableForSchemaType(item,'F1MP'),true"/>
```

Conditionally Protect Elements

The protect attribute is used to protect elements on the map based on other factors.

Syntax	Values	Description
uiHint:protect=	"'XPath','value','!=' | '='"	Used to conditionally protect this element based on the value of another element (referenced using its XPath). Enter a value of ' ' to interrogate a blank value. By default the operator is **'='**. This may be overridden using **'!='**.
	"function name, **true** | **false**"	Used to indicate a JavaScript function, which must return a Boolean.
	"'**action**','A' | 'C','!=' | '='"	Use the **'action'** setting to protect the element based on the current action. For example, certain elements may only be specified when adding a record. Any subsequent changes to the record should protect the element from being changed. When using this option, the valid values for the *'value'* are **A** (add) and **C** (change).

Warning:

Embedded spaces are not supported within the comma separated string values of this attribute.

The protect UI Hint may be used on group nodes, list nodes, and elements - except for elements within a list. Elements within a list cannot be protected conditionally.

The following UI Hint will protect the statistics category when the action is 'C'.

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
  -...
  <statisticsCategory dataType="lookup" mapField="STAT_CATEGORY_FLG" -
    lookup="STAT_CATEGORY_FLG" uiHint:protect="'action','C','='"/>
  -...
 </schema>
```

Trigger Dependent Behavior

The dependents attribute is used to trigger behavior on a child element when a parent element is changed.

Syntax	Values
uiHint:dependents=" "	A list of one or more dependent elements separated by semicolons.

Copyright © 2024, Oracle and/or its affiliates.

The following example illustrates that the dropdown list of one element is driven by the value of another element. In this example, when the Country changes, the list of States to choose from should change to only show the states for the indicated country.

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
  <country label="Country" uiHint:select="table:CI_COUNTRY" -
uiHint:dependents="state"/>
  <state label="State" uiHint:select="ss:CM-RetrieveCountryStates" -
   uiHint:selectIn="input/country:country;"  uiHint:selectOut="valuePath:output/
state/stateCode; -
   descPath:output/state/stateDesc"/>
    -...
</schema>
```

Note:

Dependent targets may only name elements, not group or list nodes.

Do not modify the "id" attribute value of dependent and parent element. Data population in dependent is done based on the "id" attribute value.

Control Rendering Target

By default all elements that are not suppressed are visible on both the display map and the input map. Use the **visibleOn** attribute to limit the inclusion of an element to either the display or input map.

Syntax	Values
uiHint:visibleOn=	**"displayMap"**
	"inputMap"

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
  -...
  <uiHint:includeMap map="StandardActionButtons" visibleOn="displayMap"
  -...
</schema>
```

Generate a Text Area

By default, a standard text box is rendered in an input map for any string element. If the field is larger and you wish to have a bigger text area (with a scroll bar), use the **textArea** attribute.

Syntax
uiHint:textArea="true"

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
   -...
  <message label="Message" uiHint:textArea="true"/>
  -...
</schema>
```

Modify FK Reference Defaults

By default, when an element with **fkRef** is displayed, an info string, context menu, navigation, and search are enabled (if the FK reference has been configured accordingly). Syntax is provided to allow you to selectively turn off any of these features.

Copyright © 2024, Oracle and/or its affiliates.

Syntax
uiHint:fkRef="info:false;context:false;navigation:false;search:false;"

Only the feature that you wish to turn off needs to be specified. The following example illustrates turning off the navigation capability, meaning the text will not be rendered as hypertext.

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
   -...
   <attachmentID fkRef="F1-ATTCH" primeKey="true" suppress="input" -
uiHint:fkRef="navigation:false;"/>
   -...
</schema>
```

Fastpath: Refer to FK Reference Formatting in the UI Map Attributes section for more information on each FK reference setting.

Suppress Automatic Number Formatting

By default numeric fields (**dataType="number"**) are formatted as numbers. An attribute is provided to instead apply alphanumeric formatting.

Note: If **dataType** is not specified explicitly, it is be derived from **mdField** or **mapField**.

Syntax
uiHint:alphaFormat="true|false"

By default, its value is **false** (and therefore can be left out altogether).

Examples:

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
  -...
  <numberCount mdField="" dataType="number" uiHint:alphaFormat="true"/>
     -...
 </schema>
```

Auto Capitalize the Input Data

The uiHint provides syntax to automatically capitalize input data.

Syntax
uiHint:capitalize="true|false"

By default, its value is **false** (and therefore can be left out altogether).

```
<schema xmlns:uiHint="http://oracle.com/ouafUIHints">
  <toDoTypeCd mdField="TD_TYPE_CD" uiHint:capitalize='true' isPrimeKey="true"/> -
</schema>
```

Note: This attribute is ignored if **uiHint:textArea="true"** is configured.

The attribute is only available in the schema designer when the **isPrimeKey** is set to **true**. The attribute may be added to any string element when using the source viewer.

Copyright © 2024, Oracle and/or its affiliates.

Schema Designer

The Schema Designer is a user-friendly interface for performing the following common schema editing tasks:

- Displaying existing schemas.
- Creating schema elements.
- Moving elements within a schema.
- Adding attribute values.

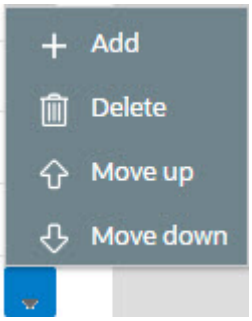
The designer provides two view modes:

- **Text** mode shows the schema elements and their attributes written in the proper syntax and allows for direct text entry.
- **Tree** mode is a view showing the elements in a tree format with many of the key attributes of each element in tabular form. While in this mode, the **Node Display** controls allow the user to choose between displaying the elements by their internal identifiers or their associated screen labels while viewing or editing the schema.

Note: Schemas that are not owned by the current installation owner are protected in both view modes.

The following sections provide more information about functionality available in the Tree mode.

- If the schema definition refers to another schema using an 'include' statement, a triangle is visible to the left of the tree area. One can expand that schema by clicking the triangle.
- Detailed information about an element's definition can be displayed and if applicable, updated from the**Tree** view by clicking the edit icon for that element, which appears on the right. Note that the element attributes are only editable if the element is defined in this schema. When viewing the attributes of an element that is part of an 'include' of another schema, the attributes are display only.
- Context-sensitive embedded help is provided for fields and controls in the edit pane by clicking the Help icon
- There is a menu dropdown icon visible to the left of the tree for any element that is defined in the current schema being viewed. It is not visible for elements included from another schema. New elements may be added in the **Tree** mode. Clicking that dropdown displays options to Add, Delete or Move an element.



Note that not all options are visible, based on what is allowed for the element adjacent to the menu. For example, on the main "schema" node, only the Add option is visible as deleting or moving the "schema" node is not applicable.

When adding a new element, you are prompted first for the element position, which is either a sibling node to the current element or a child node of that element. For either option, the new element is added below the one adjacent to where the menu is clicked. You are then prompted for the element type.

The following lists the possible element types. Most are self explanatory and represent standard schema options.

- Characteristic List. This is a special type of Flattened List element that is used to map list elements that include a sequence and a characteristic value for a given characteristic type. This element is only applicable for maintenance objects that have one or more characteristic child tables where the primary key is the maintenance object's key, characteristic type and sequence. Effective dated characteristic collections are not supported. The user defines the list name, the characteristic value element name and the characteristic type. The system will configure the remaining flattening information accordingly.

Copyright © 2024, Oracle and/or its affiliates.

- Characteristic. This is a special type of Flattened Field element that is used to map a single element to the characteristic for a given characteristic type. This element is only applicable for maintenance objects that have one or more characteristic child tables where the primary key is the maintenance object's key, characteristic type and sequence. Effective dated characteristic collections are not supported. The user defines the element name and the characteristic type. The system will configure the remaining flattening information accordingly.
- Comment. This adds a comment to the schema.
- Embedded HTML. This is specific to a schema enabled for UI Hints. It is used to include a UI map fragment.
- Field
- Flattened Field
- Flattened List
- Group
- Include BO Schema
- Include BS Schema
- Include DA Schema
- Include Map Schema
- Include SS Schema
- Input Map Title. This is specific to a schema enabled for UI Hints. It is used to define a title element for the map.
- List
- Nested Flattened Field. This is a flattened field from a child table.
- Raw Element. This element is used to capture text as is. It is typically used to capture an XML structure without any details of the definition of the individual nodes.
- Section. This is specific to a schema enabled for UI Hints. It is used to define a section within the map.
- Simple Field. This is a special type of Field element that is used to define an element that is mapped to a column that supports data defined in an XML structure. (This is either a column with the character large object data type (CLOB) or the XML data type).

The **Schema Designer** is available by choosing the **Schema** tab on the Business Object, Data Area, UI Map, Business Service, and Script pages.

Schema Viewer

The schema viewer shows a tree-view presentation of a schema in its expanded form.

The schema illustrates the structure to be used when communicating with the schema's associated object. The following takes place when a schema is expanded:

- If the schema definition includes references to other schemas, these references are replaced with the corresponding schema definitions.
- Also, if the schema definition contains **private** elements, they are omitted from this view.

Clicking on any node on the tree populates the text box on the top with the node's absolute XPath expression. You will find this feature very useful when writing scripts interacting with schema-based objects. Scripting often involves referencing elements in a schema-based XML document using their absolute XPath expression. You can use this feature on the schema viewer to obtain the XPath expression for an element and copy it over to your script.

Business Event Log

Business Event Log may be viewed as a tool designed to capture any type of business event worth noting. You configure business objects to represent the various types of events your application calls for. The following type of details may be captured for each event:

- The business object representing the type of event.
- The date and time the event took place and who initiated it.
- The business entity for which this event is logged.

Copyright © 2024, Oracle and/or its affiliates.

- Standard application message to describe the event.
- Additional context information that is available at the time of the event and varies for each type of event. The Business Event Log maintenance object supports a standard characteristics collection as well as an XML storage (CLOB) field. The event's business object determines where each piece of information resides. Refer to Business Objects for more information.

One common type of event may be the audit of changes made to sensitive data, for example, tracking an address change. Whenever an entity associated with a business object is added, changed, or deleted the system summarizes the list of changes that took place in that transaction and hands them over to **Audit** business object algorithms to process. You may design such an algorithm to audit the changes as business event logs. Refer to a business object may define business rules for more information.

You can also allow users to initiate business event logs to capture important notes about a business entity by exposing a BPA Script to invoke the event's corresponding business object.

Bottom line is that any process can create a business event log by invoking the business object representing the appropriate type of event.

Miscellaneous Topics

The following sections describe miscellaneous system wide topics.

Javadocs Viewer

The system provides a Javadocs viewer and a Groovy Javadocs viewer, which are interactive tools that allows you to browse Java documentation files (Javadocs) for Java classes that execute on the application server. For information about how to use Javadocs and navigate within the viewer, refer to Javadocs help.

Javadocs

To launch the Javadocs viewer, open open **Admin** > **Implementation Tools** > **Javadocs**.

Note: Proprietary Java Classes. A small number of Java classes have been suppressed due to their proprietary nature.

Additionally, when viewing a record that references a Java program in the algorithm type page, the batch control page or the FK reference page, an icon adjacent to Java program allows you to view the Javadocs for that program.

Groovy Javadocs

The Groovy Javadocs viewer includes the Java classes that are accessible to Groovy code within scripts. Launch this viewer using the 'View Groovy Javadocs' link in the Script Tips zone on the script page.

Module Configuration

The system provides the ability to simplify the user interface based on functionality areas practiced by your organization.

Menu items and other user interface elements are associated with function modules. By default, all function modules are accessible. If a function module is not applicable to your business you may turn it off. Refer to Turn Off A Function Module for more information on how to turn off a module.

If a function module is made non-accessible, i.e. turned off, its related elements are suppressed from the user interface. In addition the system may validate that related functionality is not accessed. This also means that turning off the wrong module may cause any of the following to occur:

- Menu items may not appear. Refer to Menu Item Suppression to better understand how menu item suppression works.

Copyright © 2024, Oracle and/or its affiliates.

- Entire menus may not appear. Refer to Menu Suppression to better understand how menu suppression works.
- Tabs on pages may not appear.
- Fields may not appear.
- The system may return an error message when you attempt to use a function (indicating the function is turned off).

To correct the above situation, simply remove the module from the turned off list thus making it accessible again.

Your module configuration setup is displayed on the installations record.

Menu Item Suppression

The following points describe how your module configuration can suppress menu items.

- Menu items that are owned by the base product (as opposed to those your implementation adds) are associated with one or more function modules. If your module configuration has turned off all of the menu item's modules, the menu item is suppressed. If at least one of the modules is accessible, i.e. turned on, the menu item is not suppressed.
- If a menu line doesn't contain any accessible items, the menu line is suppressed.
- If all lines on the Admin menu are suppressed, the menu itself is suppressed in the application toolbar. This is not expected to ever occur for the Main menu as the user preferences is on that menu and the expectation is that all users have access to their preferences.
- Menu items that are suppressed are not visible in the Menu Item Search results.

Menu Suppression

In addition to the above Menu Item Suppression logic, the following points describe how your module configuration can suppress an entire menu.

- Menus that are owned by the base product (as opposed to those your implementation adds) are associated with one or more function modules.
- If your module configuration has turned off all of the menu's modules, the entire menu is suppressed. If at least one of the modules is accessible, i.e. turned on, the menu is not suppressed.

Turn Off A Function Module

The base package is provided with a **Module Configuration** Feature Configuration that allows your organization to turn off base package function modules.

To turn off any of the base package function modules add a **Turned Off** option to this feature configuration referencing that module. Refer to the **MODULE_FLG** lookup field for the complete list of the application's function modules.

Any module not referenced on this feature configuration is considered turned on, i.e. accessible. To turn on a module, simply remove its corresponding **Turned Off** option from this feature configuration.

You may view your module configuration setup on the installation options page.

Note: Only one. The system expects only one **Module Configuration** feature configuration to be defined.

Global Context Overview

The framework web application provides each product the ability to nominate certain fields to act as a "global context" within the web application. For example, in Oracle Utilities Customer Care and Billing, the global context fields include Account ID, Person ID and Premise ID. The values of these fields may be populated as a result of searching or displaying objects that use these fields in their keys. If you navigate to the Bill page and display a bill, the global context is refreshed with the Account ID associated with that bill. The global context for Person ID and Premise ID are refreshed with data associated with that account.

The fields designated as global context for the product are defined using the lookup **F1_UI_CTXT_FLDS_FLG**.

Copyright © 2024, Oracle and/or its affiliates.

Changing the values of the global context typically cause data displayed in zones on the sidebar to be refreshed to show information relevant to the current values of these global context fields.

When the value of one of the global context fields changes, an algorithm plugged into the installation record is responsible for populating the remaining global context values accordingly. Refer to your specific product for more information about the base algorithm that is provided for that product.

System Data Naming Convention

There are several maintenance objects in the system that include owner flag in one or more of its tables. We refer to the data in these tables as "system data". Some examples of system data tables include Algorithm Type, Batch Control, Business Object and Script. Implementations may introduce records to the same tables. The owner flag for records created by an implementation is set to **CM** (for customer modification), however the owner flag is not part of the primary key for any of the system data tables. As a result, the base product provides the following guidelines for defining the primary key in system data tables to avoid any naming conflict.

Base Product System Data

For any table that includes the owner flag, the base product will follow a naming convention for any new data that is owned by the base product. The primary key for records introduced by the product is prefixed with **xn-** where **xn** is the value of the owner flag. For example, if a new background process is introduced to the framework product, the batch code name is prefixed with **F1-**.

Note: There are some cases where the hyphen is not included. For example, portal codes omit the hyphen.

For most system data, the remainder of the primary key is all in capital case. An exception is schema oriented records. For business objects, business services, scripts, data areas and UI maps, the product follows the general rule of using CapitalCase after the product owner prefix. For example, **F1-AddToDoEntry** is the name of a base product business service.

Note: Data Explorer Business Services. For business services used to invoke a data explorer zone, it is recommended to name the Business Service the same name as the related zone rather than defining a different CapitalCase name for the business service.

Please note that this standard is followed for all new records introduced by the base product. However, there are base product entries in many of these system data tables that were introduced before the naming convention was adopted. That data does not follow the naming convention described above.

Note: Schema naming conventions. A context sensitive "Schema Tips" zone is associated with any page where a schema may be defined. The zone provides recommended naming conventions for elements within a schema along with a complete list of the XML nodes and attributes available to you when you construct a schema.

Implementation System Data

When new system data is introduced for your implementation you must consider the naming convention for the primary key. The product recommends prefixing records with **CM**, which is the value of the owner flag in your environment. This is consistent with the base product naming convention. This convention allows your implementation to use the CM packaging tool in the Software Development Kit as delivered. The extract file provided with the tool selects system data records with an owner flag of **CM** and with a **CM** prefix.

Note: If you choose not to follow the CM naming convention for your records and you want to use the CM packaging tool, your implementation must customize the extract file to define the appropriate selection criteria for the records to be included in the package. Refer to the Software Development Kit documentation for more information.

Also note that owner flag may be introduced to an existing table in a new release. When this happens, the CM packaging tool is also updated to include these new system data tables. Your implementation will have existing records in those

Copyright © 2024, Oracle and/or its affiliates.

tables that probably do not follow any naming convention. After an upgrade to such a release, if you want to include this data in the CM packaging tool, you must customize the extract file for the tables in question.

Accessibility Considerations

This section provides information about topics related to accessibility that should be considered when extending the system via configuration tools.

Color Contrast

The HTML color **red** (RGB code #FF0000) when used for text on a white background does not comply with the accessibility color contrast guidelines. When it is desired to use red for text for emphasis in a zone, map or in a text string, the product recommends the shade of red with the RGB code #E0292F.

Note that in addition, the product provides a special CSS class **textColorRedOnWhite** which may be used when HTML references are used for applying the red color. For example, UI map, zone help text and scripting.

Referencing URIs

There are some configuration objects that require a reference to a URI, including file path URIs. The following sections highlight some functionality supported by the product with respect to defining / accessing URIs.

Note: In order for the functionality described below to occur, specific APIs must be used by the underlying code related to the fields that capture or process the file path or URL information. If you find that there is a URI-related field that does not provide the functionality described here, please contact customer support.

Note: For schema elements in a business object that reference a Field with the **URI** data type or define the element with the **URI** data type configured in the schema will automatically use the appropriate API that validates the value that may reference a substitution variable or must be checked against the allowlist if applicable.

Validation Against an Allowlist

Based on a property setting, your implementation may be configured to define a allowlist of URIs. If this setting is enabled, the system will issue an error if the URI is not defined in the allowlist. Consult your system administrator to verify if this setting is enabled or not for your implementation.

Fastpath: Refer to URI Substitution for information about how defining substitution tokens for URIs using the properties file technique automatically adds the defined URI to the allowlist.

URI Substitution

The system supports the ability to define substitution variables (sometimes referred to as tokens) for both URL values and file path values that reference native file storage locations. For URLs, the system supports defining variables in a Substitution Variable properties file. For native file storage paths, there are two options: the Substitution Variable properties file or via a File Storage configuration extendable lookup. More details on both these options are found below.

- Substitution variables properties file. A substitution variable for all or part of the URI definition may be configured in a properties file. This allows the system administrators to define the proper URI locations in a properties file whereas the configuration users only need to know the variable name. For example, when defining a location for an extract file in an extract batch job, instead of typing a file path of **h:\oracle\serverName\1.0.0.0\batch\extract**, the batch user can enter **@FILE_EXTRACT@**, assuming there is an entry in the substitution variables file with a name of **FILE_EXTRACT**, and a value of **h:\oracle\serverName\1.0.0.0\batch\extract**. Another example is that the batch user could enter **@BATCH_FILES@\extract**, assuming that the URI variable for **BATCH_FILES** is defined as **h:\oracle\serverName\1.0.0.0\batch**.

Copyright © 2024, Oracle and/or its affiliates.

> **Note:** The product automatically populates the value of SPLOUTPUT in the properties file so that this may be used in URI configuration. In addition, the product may supply some pre-defined variable names for other common references. As part of this, the 'advanced' menu in the system installation steps may prompt for installers to define the values of these pre-defined variables, if desired. Installations may opt to define additional substitution variables for various URI references. Refer to the *System Administration Guide* for more information.

- File Storage extendable lookup. Specifically for file paths that reference the native file system, the system also supports the ability to define a path using the **File Storage Configuration** extendable lookup. The following points highlight the steps to take for this option.
  - Navigate to Extendable Lookup and search for the **File Storage Configuration** lookup.
  - Click **Add** to create a new entry. Define a lookup value name. This will be used when configuring a file path that uses this value. Choose the File Adapter value of **Native File Storage**. Enter the desired file path value. The values defined here may in turn refer to values defined in the Substitution Variables properties file. For example, the file path can reference @SPLOUTPUT@.
  - To reference this value in system configuration, use the syntax **file-storage://XXXX**, where XXXX is the extendable lookup value. Using the same example from above, if you define an extendable lookup value of CM-FileExtract with a file path of **h:\oracle\serverName\1.0.0.0\batch\extract**, then when configuring the extract file path for an extract batch job, enter **file-storage://CM-FileExtract**. Another example is that if the extendable lookup value's file path is configured as **h:\oracle\serverName\1.0.0.0\batch**, then the user configuring the file path on the batch extract can enter **file-storage://CM-FileExtract/extract**

When should you define substitution variables in the properties file and when you define them in the extendable lookup? The following points highlight differences between the options that may help this decision.

- Substitution variables for URLs are only supported via the properties file.
- The properties file can typically only be modified by a system administrator. If there are values that are set at installation time and don't change, then defining the values in the properties file may be beneficial.
- When defining additional environments, such as test environments or production, the values in the extendable lookup may be copied using CMA. Ideally the values are defined such that they are the same between various regions.

External File Storage

The system supports using Oracle Cloud Object Storage for managing files in cloud implementations and can be configured to read files from or write files to this external location. Refer to Oracle Cloud Object Storage documentation for more information about obtaining an account and defining appropriate file locations or "buckets".

Once your cloud storage information is defined, the following points highlight the configuration steps required in the application.

- Define a signature key ring. When the system tries to communicate with cloud object storage, it must provide a signature key so that cloud storage can confirm that the request is from a trusted source.
  - Navigate to Key Ring in add mode and select the **RSA Signature Key Pair** business object.
  - Define a key ring code, which will be used in the File Storage configuration (below) along with a description.
  - Once the key ring is added, click **Generate Key** to generate a private / public key pair.
  - Click View Public Key to launch a pop-up that displays the public key, allowing the user to copy the key.
- At this point, the user should navigate to cloud object storage and register the public key. Refer to Cloud Services Administration Guide for more information on this step. Once this is done, the key ring is now ready to be configured in file storage configuration.
- Next, confirm that the region where your object storage is defined is configured in the system and if not, define the value.
  - Navigate to Extendable Lookup and search for the **OCI Region** lookup. Review the list of regions already provided.
  - If your region is not listed, click the Add button. Be sure that the Region Key and the Region Identifier match the values defined in the OCI region documentation. The region key is a three digit identifier. For example, the one for Sydney region in Australia is SYD. The region identifier follows a pattern of a two digit realm id followed by

Copyright © 2024, Oracle and/or its affiliates.

the city name, followed by a single digit, all separated by hyphens. For example, the one for the Sydney region is "ap-sydney-1".

- Use a File Storage extendable lookup to define the location and connection information so that you may reference this location in system configuration. The following points highlight the steps to take for this option.
  - Navigate to Extendable Lookup and search for the **File Storage Configuration** lookup.
  - Click **Add** to create a new entry. Define a lookup value name. Note that the lookup value should not have a slash or backslash in its name. This will be used when configuring a file path that uses this value. Choose the File Adapter value of **Oracle Cloud Object Storage**. Provide the following information that identifies the cloud storage options: User, Tenancy, Compartment, Namespace and Region. For the Key Ring, choose the value defined above.
- To reference this value in system configuration, use the syntax **file-storage://XXXX/...**, where XXXX is the extendable lookup value and any additional path information that is appropriate. For example, if you define an extendable lookup value of **CM-CloudStorage** and you have a bucket defined in Cloud Storage for processUpload, when configuring the file path for the upload batch job, enter **file-storage://CM-CloudStorage/processUpload**.
- The product supports referencing virtual folders for a bucket. Simply add the virtual folder name(s) after the bucket name. For example, if there is a bucket called 'financials' and a virtual folder for 'paymentUpload', the file path may have this value: **file-storage://CM-CloudStorage/financials/paymentUpload**. For extract processes, if the file path references a virtual folder that does not exist for the referenced bucket, the virtual folder is created as part of the extract process.
- You may optionally use a Bucket Name Prefix on the file storage configuration to support defining different Bucket Names for different environments. For example, imagine you want to have a separate "processUpload" bucket for a development region, a test region and a production region. In the Cloud Storage configuration you can define separate buckets for "DEV_processUpload", "TEST_processUpload" and "PROD_processUpload". If you needed to define the full bucket on each file path reference on various batch jobs, migrating the batch job configuration from one environment to another would require a manual step to change all the bucket references. Instead, the system supports defining a Bucket Name Prefix, which if populated will be prepended to bucket reference before connecting to Cloud Storage. This means that the prefix is defined once and all the references throughout the system can keep the common part of the Bucket name (such as "processUpload"). Then only the File Storage configuration needs to differ between the different regions.
- If your implementation integrates with an external reporting tool that can write the report output to specific delivery channels (buckets) check the Reporting Configuration switch. Define one or more Bucket Names for the buckets that the reporting tool may use for delivery channel options.

Note: Refer to the Key Ring documentation for information about practicing key rotation.

Caching Overview

A great deal of information in the system changes infrequently. In order to avoid accessing the database every time this type of information is required by an end-user or a batch process, the system maintains a cache of static information on the web server and in the batch thread pool worker. These are referred to as the "application caches". Some examples of application caches include

- System messages
- Field label and other field information
- Security Information

The framework product provides many specific caches for commonly used (and infrequently changed) data. In addition, specific edge applications may introduce additional caches as appropriate.

Information may also be cached on each user's browser.

The following topics highlight information about refreshing the various caches.

Copyright © 2024, Oracle and/or its affiliates.

Server Cache

The server cache refers to data that is cached on the web server. An important use of this cache is for users' online access to the application. The caches aid in better performance while navigating throughout the system, allowing for data to be accessed from the cache rather than by always accessing the database. Besides user access to the web server cache, other functionality deployed to the web server uses caches in a similar way. For example, web services are deployed to the web server and access their own version of the cache.

The contents of the cache are cleared whenever the web server is restarted. This means that fresh values are retrieved from the database once users and web services start using the application again.

The product also supplies a flush command that one can issue in the browser's URL to immediately clear the contents of the cache. The command **flushAll.jsp** flushes every cache.

For example, assume the following:

- the web server and port on which you work is called **OU-Production:7500**
- you add a new record to a control table and you want it to be available on the appropriate transactions immediately

You would issue the following command in your browser's address bar: **http://OU-Production:7500/flushAll.jsp**. Notice that the command replaces the typical cis.jsp that appears after the port number.

If your system has been configured correctly, the **flushAll** command will submit a request to do a "global" flush of caches (including the web services cache and the thread pool worker cache). This functionality uses a JMS Topic to publish the flush request. Refer to the *Server Administration Guide* for details on how to configure the JMS topic.

Also note that the system provides a business service **F1-FlushAllService**, which is equivalent to the flushAll command (except that it does not restart your browser session). This may be useful for any processes that are built to orchestrate updates to configuration data where you want to ensure that the cache is immediately flushed. This business service is also exposed as a REST web service.

Batch Cache

When submitting a batch job, the batch component uses a Hibernate data cache to cache administrative data that doesn't change very often. The tables whose records are included in this cache are configured using the Caching Regime value of **Cached for Batch**. This is often referred to as the "L2 cache". Refer to Tables for more information. When starting a thread pool worker, data in tables marked as cached is loaded and cached for as long as that thread pool is running.

In addition batch jobs may also access application caches when applicable. When starting a thread pool worker, application data that is cached is loaded and cached for as long as that thread pool is running.

If there are any batch jobs that update any tables that are marked as **Cached for Batch**, you must run that batch job using a special thread pool that is configured with the caching disabled. Contact your system administrator to confirm the thread pool that is configured this way.

If there is a change in cached data that should be available for the next batch job, the following points highlight how the cache can be refreshed:

- By default the system is configured to automatically refresh the Hibernate cache every 60 seconds. However, an implementation may override the configuration to either change the number of seconds between intervals or to disable the automatic caching altogether. Application caches used by the batch jobs are not impacted by this refresh.
- Restart the thread pool workers.
- Run the **F1-FLUSH** (Flush all Caches) background process. This background process will flush the application data cached for all thread pool workers for all thread pools.
- If your the region has configured the thread pool workers to "listen" to requests for global flush as described in the Server Cache section, the thread pool worker caches are also refreshed when a **flushAll** command is issued.

Client Cache

In addition to the web server's cache, information is also cached on each user's browser. After clearing the cache that's maintained on the web server, you must also clear the cache that's maintained on your client's browser. To do this, follow the following steps (note that depending on the browser you use, the steps may vary slightly):

Copyright © 2024, Oracle and/or its affiliates.

- Select **Tools** on your browser's menu bar
- Select **Internet Options...** on the menu that appears.
- Click the **Delete Files** button on the pop-up that appears.
- Turn on **Delete all offline content** on the subsequent pop-up that appears and then click **OK**.
- And then enter the standard URL to re-invoke the system.

Note: Automatic refresh of the browser's cache. Each user's cache is automatically refreshed based on the **maxAge** parameter defined in the web.xml document on your web server. We recommend that you set this parameter to **1** second on development / test environments and **28800** seconds (8 hours) on production environments. Please speak to system support if you need to change this value.

Expression Parser

The product provides support for defining expressions that may be of a mathematical or logical/boolean nature. The expression may include variables and functions.

The data explorer column parameter is an example of where this may be used. That parameter supports the definition of a formula. Edge applications may include support for a formula or expression using this parser as well. For example, several application include a type of 'rule' object (calculation rule, form rule or usage rule) that is used for validation or calculation that may support applying a formula.

The following tables highlight what is supported in the expressions that use this parser.

Category	Supported in Expression	Description
Data types	Number	
	String	
	Boolean	
	List	
Literals	Numbers	
	Strings surrounded with either single quote or double quote.	
	Note: 'Escaping' special characters is not currently supported.	
	Boolean values: **true** and **false**.	
Operations	+	Plus
	—	Minus
	/	Division
	*	Multiplication
	^ or **	Power
	%	Modulus
Logical operations	=	Equal
	>	Greater than
	>=	Greater than or equal t
	<	Less than
	<=	Less than or equal to

Copyright © 2024, Oracle and/or its affiliates.

Category	Supported in Expression	Description
	!= or <>	Not equal to

This table identifies the functions that are supported. Note that several of the functions are applicable to a list of values. Note that although the functions are listed in lower case, the column parameter syntax in data explorer indicates referencing the functions as all capital letters. The system converts the data explorer column formula to lowercase before being evaluated.

Function	Parameter	Results
size( )	List element	Number of elements in the list.
isEmpty( )	List element	Returns true if the list is empty.
sum( )	List element of type 'number'	Returns the sum of the numbers
avg( )	List element of type 'number'	Returns the average of the numb
	One or more numbers separated by commas	Returns the average of the numb
max( )	List element	Returns the largest value in the
	One or more comparable elements.	Returns the largest value of the
min( )	List element	Returns the smallest value in the
	One or more comparable elements.	Returns the smallest value of the
abs( )	Number	Returns the absolute value.
ceiling( )	Number	Rounds the number to the ceilin
exp10( )	Number	Raises 10 to the number power.
acos( )	Number	Returns the arc cosine of the nu
asin( )	Number	Returns the arc sine of the numb
atan( )	Number	Returns the arc tangent of the nu
cos( )	Radian	Returns the cosine of the radian
exp( )	Number	Raises e to the number power.
log10( )	Number	Takes the log, base 10, of the nu
log( )	Number	Takes the natural log (base e) of
sin( )	Radian	Returns the sine of the radian ar
sqrt( )	Number	Returns the square root of the nu
tan( )	Radian	Returns the tangent of the radian
floor( )	Number	Rounds the number to the floor.

Copyright © 2024, Oracle and/or its affiliates.

Function	Parameter	Results
round( )	Number	Assumes a scale of 0. The defa half up" is applied.
	Number, Scale	The default rounding mode of "
	Number, Scale, Mode	The mode must be set to one of • **"ROUND_CEILING"** • **"ROUND_DOWN"** • **"ROUND_FLOOR"** • **"ROUND_HALF_DOWN"** • **"ROUND_HALF_UP"** • **"ROUND_HALF_EVEN"** • **"ROUND_UP"** • **"ROUND_UNNECESSAR**
negate( )	Number	Returns the negative value of th

The following are special functions supported in the application for a list of values. In each case, the syntax is *function* [*indexVariable* in *listName* | *expression using indexVariable*], where the *indexVariable* is chosen by the formula writer to represent each entry in the list and the expression used to evaluate each entry must reference that variable.

Note: The syntax supported for a given use of the formula in a functional area is driven by that particular functional area. For example, in Oracle Public Sector Revenue Management, a formula is supported in the "conditional element validation" form rule. In that form rule all variables including lists are declared in the form rule using letters and the formulas in turn use these letters. In that scenario, the functions below would reference the declared variable letter as the "listName". Other specific functional area that use this expression parser may support different syntax for referencing elements or lists.

Function	Description	Examples
any [ ]	This function returns the value **true** if any of the entries in list satisfies the expression.	The following `any [ i in`
all [ ]	This function returns the value **true** if all of the entries in the list satisfy the expression.	The following `all [ i in`
collect [ ]	This function returns a new list of elements from the referenced list where the value of each entry of the new list is the result of the expression applied to each original value.	The following `collect [ i`
select [ ]	This function returns a list of all the values of the original list that satisfy the Boolean expression.	The following `select [ i`
reject [ ]	This function returns a list of all the values of the original list that do not satisfy the Boolean expression.	The following numbers. `reject [ i`

Copyright © 2024, Oracle and/or its affiliates.

Debug Mode

Your implementation team can execute the system using a special mode when they are configuring the application. To enable this mode, enter **?debug=true** at the end of the URL that you use to access the application. For example, if the standard URL was **http://CD-Production:7500/cis.jsp**, you'd enter **http://CD-Production:7500/cis.jsp?debug=true** to enable configuration mode.

In addition, if you have security access to the application service **F1OUDBUG**, you can enable this mode without changing the URL via the Help Menu - Enable Debug option. If debug mode is enabled, the help menu entry shows a Disable Debug option, allowing you to turn it off without changing the URL.

When in this mode certain debugging oriented tools become available via an icon visible. Clicking the icon displays a slide-out panel with options defined below.

The following are only applicable in a non-Cloud environment:

- **Start Debug** starts a logging session. During this session the processing of various configuration tools objects are logged. For example, the log will show the data areas that are passed in at each step and the data areas returned after the step is processed.
- **Stop Debug** stops the logging session.
- **Show Trace** opens a window that contains the logging session. All of the steps are initially collapsed.
- **Clear Trace** clears your log file.

Note: The buttons are secured. An application service **F1CONFIGLOG** has been provided for this functionality to allow implementations to restrict user access to these buttons. An additional application service **F1DEBUG** is checked when the environment is not a 'development' environment. This allows implementations to provide an additional layer of security for their production users.

The following are available to view log entries captured while using the system.

- **Show User Log** allows you to view your own log entries. The number of "tail" entries to view may be specified in the adjacent **Log Entries** field before clicking the button. Limiting the number of entries to view allows the user to quickly and easily see only the latest log entries without having to manually scroll to the end of the log.
- **Show Log** allows for advanced settings. It provides for a date / time range to allow a user to target the output for a specific period. In addition, administrative users may request to view logs of other users. The following are examples of when this capability is needed:
  - Batch and integration logs are captured by special user codes created for this type of processing. These users typically do not represent a human and therefore wouldn't access logs using "show user log". Instead someone else would be accessing these logs.
  - Customer admins would like to retrieve logs for an end user possibly for support reasons. An end user might not have technical skills to gather or assess the logs nor would they be given the privilege to access the logs for security reasons
- Checking the **Global Debug** indication starts various tracing options that are added to the user log.

Note: Show User Log button is secured. An application service **F1USERLOG** has been provided for this functionality to allow implementations to restrict user access to this button. Only users with access to the **Administrative** access mode of the application service are allowed to view logs of other users. Such restrictions may be called for in production environments.

For Cloud implementations only, the following buttons are available in this mode:

- **Restart TPWs** - This button forces batch containers to terminate, interrupting all running batches. New containers will spin up and the interrupted batches resume. Refer to the Batch Best Practices technical reference paper for more information about this button and when to use it.
- **Restart CM Jar Servers** - This button is provided to aid with support in the cloud for custom Java code in certain use cases. Refer to the Cloud Services Implementation Guide for more information.

Copyright © 2024, Oracle and/or its affiliates.

Other parts of the system may show additional configuration oriented icons when in this mode. For example, explorer zones may provide additional tools to assist in debugging zone configuration. These icons are described in the context of where they appear.

Also, in debug mode drop down lists in data explorer and UI map zones will contain the code for each item in addition to the item's display string.

Prepare Issue Details

When a user encounters an error with the product and submits a service request, information about the environment, the release and the configuration of the user is often very helpful in understanding the cause of the problem. The system provides a script to capture the environment details and issue description for reporting purposes.

To access the script use **Menu** > **Tools** > **Prepare Issue Details**. The option is also available in the help menu in the toolbar.

A pop-up window is provided to display the details that are being captured, including a generated Issue ID to uniquely identify the report. The pop-up allows the user to add additional information describing the specific issue. When the user presses Save, the details are added to the system log with the issue ID, which can be searched when investigating the problem. A message will be displayed to either confirm the details were successfully logged or indicate that an error occurred.

Note: The script and menu item are secured. An application service **F1-PREPISSUEDTLS** has been provided to allow implementations to restrict user access to this functionality.

System Override Date

The system provides a way to override the system date used for online operations. This feature is available if the server administrator has enabled it in the environment properties. For instructions on configuring environment properties see the *Server Administration Guide*. The system date override feature is not recommended for production environments.

Under the **General System Configuration** Feature Configuration, the **System Override Date Option Type** holds the date the application will use as the global system date instead of retrieving the same from the database. This feature can be especially useful in running tests that require the system date to be progressed over a period of time.

The system override date feature is also available at the user level. This is useful when a user wants override the system date to run tests without affecting the system date for other users in the environment. In order to override the system date for the user, open the User - Characteristics page, add the**System Override Date** characteristic type with a characteristic value set to the desired date in the YYYY-MM-DD format.

If system override dates are defined at both the feature configuration level and the user level, the date set at the user level will take precedence.

Additional Search Options

The product supports fuzzy searching in explorer zone types using the Oracle Text CONTAINS operator.

Refer to the DBA guide for details on setting up the database to support fuzzy searching. Note that there are some implementations where fuzzy searching will not be possible. For example, it's only available for implementations using the Oracle database. Additionally, not all languages are supported. Refer to the Oracle Database documentation for more information about fuzzy searching.

For information about the particular syntax to use in the explorer zones, refer to SQL Statement in the zone parameter details section.

Copyright © 2024, Oracle and/or its affiliates.

Extensions Dashboard

This portal provides a high level summary of configuration entities that have been customized and extended to meet your business requirements. Included are customer owned entities and extended product owned entities. The portal is intended to be used by your implementation team to get an appreciation of areas that have been customized and may need to be reviewed or retested for various reasons.

Use **Admin** > **Implementation Tools** > **Extensions Dashboard** to navigate to the portal.

The following zones may appear as part of the portal's **Main** tab page

- **Extensions Summary.** This zone provides a summary of customized entities by maintenance object. You may select a specific maintenance objects to review entities of this type.
- **Extension Entities.** This zone lists customized entities of the selected maintenance object. It is only visible when a maintenance object is selected on the Extensions Summary zone.
- **Extension Details.** This zone provides additional information as to which specific tables contain the custom owned content for the selected entity. It is only visible when a specific customized entity is selected.
- **Revision Control.** This zone lists revisions made to the selected entity. It is only visible when revision information exists for the selected entity.
- **Configuration Migration.** This zone lists CMA migration objects that applied changes to the selected entity. It is only visible when import information exists for the selected entity.

The following zone may appear as part of the portal's **Review** tab page

- **Duplicate Configurations.** This zone highlights duplicate configuration issues that may rarely arise after an upgrade. Customers should use this information to adjust thir configuration accordingly.

Chapter 6

To Do Lists

Topics:

- The Big Picture of To Do Lists
- Setting Up To Do Options

Certain events that occur within the system will trigger messages describing work that requires attention. For example, if a bill segment has an error, the system generates a To Do message to alert the person responsible for correcting such errors.

Each type of message represents a To Do list. For example, there are To Do lists for bill segment errors, payment errors, customer contact reminder, etc.

We refer to each message as a **To Do Entry**. Each To Do entry is assigned a specific **To Do Role**. The role defines the users who may work on the entry. A To Do entry has a **To Do log** that maintains record of the progress on the To Do entry. For example, the To Do log indicates when the To Do entry was created, when it was assigned to a user and to whom it was assigned, and when and by whom it was completed.

Fastpath: Refer to To Do Processing for a description of end-user queries and tools assisting in reviewing, assigning and processing To Do entries.

The Big Picture of To Do Lists

The topics below provide more information about To Do configuration.

- To Do Entries Reference A To Do Type on page 372
- To Do Entries Reference A Role on page 372
- The Lifecycle Of A To Do Entry on page 373
- How Are To Do Entries Created? on page 374
- Refining a To Do Entry's Details at Creation Time on page 377
- The Priority Of A To Do Entry on page 379
- What Happens When You Work A To Do Entry on page 380
- Monitoring A To Do Entry on page 380
- To Do Entries Have Logs on page 380
- Implementing Additional To Do Entry Business Rules on page 381
- To Do Entries May Be Routed Out Of The System on page 381
- To Do Information May Be Formatted By An Algorithm on page 381
- Periodically Purging To Do Entries on page 381

To Do Entries Reference A To Do Type

Every To Do entry references a To Do type. The To Do type controls the following functions:

- The page into which a user is taken when they drill down to the related object on an entry.
- The message associated with the To Do that appears on various pages. Note this message can be overridden for specific To Do messages by specifying a different message number in the process that creates the specific To Do entry. For example, the process that creates To Do entries associated with bill segments that are in error displays the error message rather than a generic "bill segment is in error" message.
- The To Do entry sort keys. Note that the processes that create To Do entries are responsible for populating the sort key values.
- Whether (and how) the To Do entry is downloaded to an external system (e.g., an email system).
- The roles to which an entry may be reassigned.
- The default priority of the To Do entry. Note that this value may be overridden by a **Calculate Priority** algorithm.
- An indication of whether a To Do of that type may be created manually by a user.
- The algorithms used to perform specific business rules for To Do entries of this type.
- The characteristics applicable to the To Do.

To Do Entries Reference A Role

Every To Do entry references a role. The role defines the users who may be assigned to **Open** entries.

The permissible roles that may be assigned to a To Do entry are defined on the entry's To Do type. After an entry is created, its role may be changed to any role defined as valid for the entry's To Do type.

The base package determines the initial To Do Role as follows:

- The system checks if an entry's message category / number is rerouted to a specific role. If so, it defaults this role. Refer to Adjusting the To Do Role for more information.
- There may be some other condition related to the object linked to your To Do entry that warrants an override role. A To Do pre-creation plug-in would be used to check the conditional logic and override the role, if applicable.
- If a Role wasn't determined in one of the previous steps and a Role is provided by the initiating process (algorithm or background process), the entry is created with that Role.
- If the entry does not have a role after the above takes place, the entry's To Do type's default role is assigned to the entry.

Copyright © 2024, Oracle and/or its affiliates.

> **Note:**
>
> At installation time, the system provides a default role assigned to the system To Do types when first installed called **F1_DFLT**. This is done to allow testing of the system prior to implementing of appropriate To Do roles for your organization. The recommendation is to configure all the To Do Types with appropriate business oriented To Do roles once they are defined.

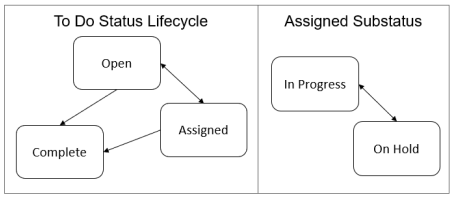
Supervisors for a Role

Most organizations have the notion of a supervisor who is responsible for all entries assigned to a given role. A user that is considered a supervisor should be a valid user in a role. In addition, application security is used to provide supervisors access to additional capabilities, namely the ability to review To Do entries assigned to other users and the ability to assign To Do entries to other users.

The pages listed in To Do Supervisor Functions are meant for supervisors only. So security for these pages should be restricted to those users that are considered supervisors. The To Do Management portal and the To Do Search page are available to all users. There is a special access mode for Supervisors to allow those users the ability to assign To Do entries to other users.

The Lifecycle Of A To Do Entry

The following state transition diagram will be useful in understanding the lifecycle of a To Do entry.



A To Do entry is typically created in the **Open** state. Entries of this type can be viewed by all users belonging to the entry's role. Refer to How Are To Do Entries Created? for information about how entries are created.

An **Open** entry becomes **Assigned** when it is assigned to a specific user or when a user proactively assumes responsibility for the entry. While an entry is **Assigned**, it is visible on the To Do Summary page only by the user who is assigned to it.

> **Note: To Do entries may be created in the Assigned** state. Some To Do background processes may create To Do entries in the **Assigned** state. When a user adds a To Do entry online and assigns the To Do to a user (as opposed to a role), the To Do entry is also created in the **Assigned** state.

An **Assigned** entry may be forwarded to a different user or role. If the entry is forwarded to a role, it becomes **Open** again. This may also be referred to as "reopening" or "unassigning" a To Do entry.

While a To Do Entry is in the **Assigned** state, if your implementation has configured it, you may further indicate the substatus of **In Progress** or **On Hold**.

- When you are ready to start work on an entry you can put it in the **In Progress** state. Refer to Working On A To Do for more information on how to do this.
- If you are not able to complete the To Do entry, you can put it **On Hold**, or stop working on it to put it back to the **Assigned** state (basically resetting the substatus). In either case, the To Do entry is still assigned to you.
- You may only have one To Do entry in the **In Progress** state. If you already have an entry in this state and you perform any action that puts a new To Do entry in **In Progress**, the other To Do entry will automatically be put into the **On Hold** state.

Copyright © 2024, Oracle and/or its affiliates.

> **Important:** To configure the system to allow users to use the substatus capability, you simply need to give users security for the Start Progress and Hold Progress and Stop Work access modes on the To Do Entry application service (**CILQTDEP**). The recommendation is that an implementation either grants all To Do users access to these access modes or grants no users access to these access modes.

> **Note:** The substatus values are captured in a separate field (**TD_SUB_STATUS_FLG**). They are only populated if the main status field (**ENTRY_STATUS_FLG**) is in the Assigned status (**W**). But most pages in the system treat these substatus values as another status value. You typically won't see both values Assigned and In Progress or Assigned and On Hold. You'll just see In Progress and On Hold.

When an entry becomes **Complete**, it is no longer visible in various To Do queries (but it remains on the database for audit purposes). There are two ways an entry can become **Complete**:

- A user can manually indicate it is **Complete** (there are several ways to do this).
- For To Do entries that are logically associated with the state of some object, the system automatically marks the entry **Complete** when the object is no longer in the respective state. For example, an entry that's created when an account doesn't have a bill cycle is completed when the account has a bill cycle.

  > **Important:** The automatic completion of To Do entries occurs when the background process responsible for creating entries of a given type is executed. Therefore, if you only run these processes once per day, these entries remain **Assigned** even if the object is no longer in the respective state.

- The To Do monitor process may be used to evaluate whether or not the To Do entry can be closed automatically. Refer to Monitoring a To Do Entry for more information.

How Are To Do Entries Created?

A To Do Entry may be created in the following ways:

- A background process can create To Do Entries.
- An algorithm can create entries of a given type. Because the use of algorithms is entirely dependent on how you configure the control tables, the number of types of such entries is indeterminate.
- A user can create entries of To Do types that have a **Manual** usage. Refer to To Do Entries Created Manually for information about setting up manual To Do types.

For any base product process that includes logic to create a To Do entry, the system supplies a sample To Do type that may be used. Although the To Do types provided by the product are system data, the following information related to each To Do type may be customized for an implementation and is not overwritten during an upgrade:

- The creation process. If the To Do is created by a background process where the background process is referenced on a To Do type. Refer to To Do Entries Created By Background Processes for more information.
- The routing process. Refer to To Do Entries May Be Routed Out of the System for more information.
- The priority. Refer to To Do Type - Main for more information.
- The roles that may be associated with the To Do type. Refer to To Do Entries Reference a Role for more information.
- The message override information. Refer to Suppress the Creation of the To Do Entry, Adjusting the To Do Role and What Happens When You Work A To Do Entry for more information.

- Suppress the Creation of the To Do Entry
- To Do Entries Created By Background Processes
- To Do Entries Created By Algorithms
- To Do Entries Created Manually

Suppress the Creation of the To Do Entry

The system provides the ability to indicate that a To Do Entry should not be created based on conditions. When the system detects that no To Do should be created, there is no error. The system simply returns to the caller with an indication that

Copyright © 2024, Oracle and/or its affiliates.

no To Do was created. This functionality only applies to automatic To Do Entries created by an algorithm or background process. It does not apply to manually created To Do Entries.

Suppress Based on To Do's Message

There are some types of To Do Entries that are generated with a message that is specific to a condition that was found for a record. For example, business errors found when trying to update a record through a system process will generate a To Do entry using the error message as the To Do Entry's message. There may be some use cases where the error reported is not something that a user needs to address and it may be beneficial to your organization to suppress the To Do Entry. Note that this is only appropriate if the message generated is for informational or warning purposes and doesn't require action or if you know that the error will be resolved by a subsequent step.

The configuration to suppress a To Do entry based on its message is on the To Do type's message overrides page.

Note: Message Category / Message Number. Every error in the system has a unique message category / number combination. Refer to The Big Picture of System Messages for more information about message categories and numbers.

We do not supply documentation of every possible message that can be handled by a given To Do type. The best way to build each To Do type's reroute list is during the pre-production period when testing the system. During this period, compile a list of the messages that warrant special behavior described here.

Note that if the To Do type is configured to reference a specific message category / number, it means all To Do entries have the same message. If your organization does not want To Do entries created for this To Do type, then rather than using this suppression technique, you should identify the background process or algorithm that generates the To Do entries of this type and change the configuration there such that it does not try to create the To Do in the first place.

Suppress Based on a Condition

There may be use cases where a To Do entry should be suppressed based on some conditional logic. For example, consider the billing process in Oracle Utilities Customer Care and Billing. There is typically a bill cycle schedule that allows for attempting to bill all the accounts for the bill cycle over a multi-day window. There are some errors that occur on day 1 of the window that may be resolved by day 2 or 3. An implementation may choose to suppress To Do entries for this condition if this is day 1 or day 2 of the bill cycle. If the error still occurs on day 3, then the To Do entry should be generated. This type of condition needs to be determined using an algorithm. The To Do pre-creation plug-ins support returning an indication to suppress the To Do entry. Refer to To Do Pre-creation Plug-ins for more information.

To Do Entries Created By Background Processes

There are different types of To Do entries created by background processes:

- To Do entries created by dedicated To Do background processes
- To Do entries created for object-specific errors detected in certain background processes
- To Do entries created based on a specific condition

- Dedicated To Do Background Processes
- To Dos Created for Object-Specific Error Conditions
- To Dos Created by Background Processes for Specific Conditions

Dedicated To Do Background Processes

There are To Do entries that are created by system background processes whose main purpose is to create To Do entries based on a given condition. For these background processes, the To Do Type indicates the creation background process.

Note that the Plug-in Driven batch program cannot be used for this purpose. This program assumes that any To Do Type that references it as a creation process is using the link for the Object-Specific Error Condition use case. If you want to use the plug-in driven batch process to create To Do entries based on a given condition, then you need to follow the pattern described in To Dos Created by Background Processes for Specific Conditions.

Copyright © 2024, Oracle and/or its affiliates.

Note: If you don't schedule the background process, the entries will not be created! The To Do entries of this type will only be created if you have scheduled the associated background process. Therefore, if you want the system to produce a given entry, schedule the background process.

To Dos Created for Object-Specific Error Conditions

A system background process may create a To Do entry when an error is detected during object-specific processing. This is applicable for processes that do not have built in error handling, for example where there is an explicit "error" state or where the record has an explicit "exception" record.

For these background processes, the To Do Type must reference the creation background process.

To have the system create To Do entries for some or all of the errors generated by one of these processes, you must do the following:

- If you want the system to generate To Do entries for errors detected by one of the background processes below, go to the appropriate To Do type and populate the creation background process.
- If you want the system to generate To Do entries for some errors for the process, but not for all errors, populate the creation background process and then proceed to the message overrides tab to suppress certain messages. Do this by indicating the message category and message number you want to suppress. Any error that is suppressed is written to the batch run tree.

The functionality will only create a new To Do entry if there is not already an existing (non-complete) To Do for the same To Do type, drill key and message category / message number. It will also check for an existing To Do for a successfully processed record and complete that To Do.

If you do not populate the creation background process, the errors are written to the batch run tree.

Note: Errors received while creating a To Do entry. If the background process cannot successfully create a To Do entry to report an object-specific error, the error is written to the batch run tree along with the error encountered while attempting to create the To Do entry.

Note: System errors are not included. To Do entries are not created for a system error, for example an error related to validation of input parameter. These errors are written to the batch run tree. Refer to Processing Errors for more information.

To Dos Created by Background Processes for Specific Conditions

There are some system background processes that create a To Do entry when the process detects a specific condition that a user should investigate. For each background process, the To Do type is an input parameter to the process. The system provides To Do types for each base package background process that may create a To Do entry.

Note: No Creation Process. These To Do types do not need (and should not have) a **Creation Process** specified.

To Do Entries Created By Algorithms

There are To Do entries that are created by algorithm types supplied with the base package. The system supplies a To Do Type for each of these To Do entries that you may use.

If you want to take advantage of these types of entries for system algorithm types, you must do the following:

- Create an algorithm:
  - This algorithm must reference the appropriate Algorithm Type.
  - These algorithms have a parameter of the To Do Type to be created. You should specify the To Do Type indicated in the table.
- Plug the algorithm into the respective control table.

Copyright © 2024, Oracle and/or its affiliates.

To Do Entries Created Manually

You must set up manual To Do entry types if you want your users to be able to create To Do entries online. Users may create a manual To Do entry as a reminder to themselves to complete a task. Online To Do entries may also be used like electronic help tickets in the system. For example, if a user is having a problem starting service, the user can create a To Do that describes the problem. The To Do can be assigned to a help resolution group that could either resolve the problem or send the To Do back to the initiating user with information describing how to resolve the problem.

If you want to take advantage of manual To Do entries, create a To Do type and specify the following information.

On the Main tab:

- Set the **To Do Type Usage** flag to **Manual**.
- Set the **Navigation Option** to **toDoEntryMaint** (To Do entry maintenance).
- Set the **Message Category** and **Message Number** to the message you want to be used for To Do entries of this type. The system will populate the message parameter with the Subject. To show only the subject in the To Do's message, use a message with "%1" as its text.

On the Roles tab:

- Specify the To Do roles that may be assigned to To Do entries of this type.
- Indicate the To Do role that should be defaulted when you create To Do entries of this type.

On the Sort Keys tab:

When a user adds a manual To Do entry, the system creates an entry with three sort key values. (Sort keys may be used on the To Do list page to sort the entries in a different order.) The To Do type should be set up to reference the sort keys as follows:

Sequence	Description
1	Created by user ID
2	Created by user name
3	Subject

We recommend that the keys have an **Ascending** sort order and that the Subject is made the default sort key.

Note: It is possible to define additional sort keys and use a To Do Post Processing algorithm to populate the values. In this case, the base sort keys defined above should still be defined.

On the Drill Keys tab:

When a user adds a manual To Do entry, it is created with a drill key value equal to the To Do entry's ID. When the user clicks the Go To button next to the message in the To Do list, the system uses the drill down application service (defined on the main tab) and the drill key to display the associated To Do entry.

The To Do type must be set up with a drill key that reference the To Do entry table and the To Do entry ID:

Sequence	Table	Field
1	CI_TD_ENTRY	TD_ENTRY_ID

Refining a To Do Entry's Details at Creation Time

When a To Do Entry is created, the key information provided to the system are the To Do Type and the drill key, which is the identifier of the object related to the To Do entry. With this information, system logic along with algorithms called at creation time can add details to the To Do entry or update important attributes so that the To Do can be worked on by the right team in the right priority.

The topics in this section will highlight the features provided to further refine a To Do Entry at creation time.

Copyright © 2024, Oracle and/or its affiliates.

Linking Additional Information to a To Do Entry

Additional information may be linked to a To Do entry using characteristics. This may include master data so that the To Do entry is associated with one or more specific main objects in the system. It may also include other information that is helpful to any user that may be working on the To Do entry.

For **manual** To Do entries, the valid characteristic types that may be linked to the To Do entry must be defined on the To Do Type for that To Do entry. When create a manual To Do entry, the user populates the related characteristics as part of creating the To Do.

For **automatic** To Do entries, the program that generates the To Do may link related data to the To Do using characteristics.

In addition, the system To Do Pre-creation plug-ins can be used to link additional characteristics. These algorithms would use the drill key, which is the related object for the To Do entry. Based on the drill key, it can look up other information and create appropriate characteristics. Your specific product may be providing one or more algorithms to determine key master data records associated with the related object. For example, Oracle Utilities Customer Care and Billing provides an algorithm that attempts to link a related person, account, premise, service agreement and service point for that drill key. Other edge applications provide a similar algorithm for their relevant entities.

Any information that your user may need to have to help work a To Do entry should be linked to the To Do via a characteristic. For example, if the Rate Class of an account is important or the amount of the previous bill will help determine the priority of the To Do entry, this information may need to be linked as a characteristic.

Note: With the volume of To Do entries in the system, it is recommended to carefully consider what information you are linking to the To Do as a characteristic. Users can drill down on details when investigating a To Do entry. Only very important and critical information that would help users in reviewing To Do entries in the To Do Management portal and make decisions about prioritization and assignments should be linked as characteristics.

Note, before you can set up these algorithms, you must define the characteristic types that you'll use to hold each of these entities. Also note that it is not necessary to define these characteristics as valid characteristic types on the To Do type.

If your To Do entries reference characteristics that are related to your global context data, you may want to configure an alert algorithm to display an alert if a related entry is **Open** or **Assigned**. Refer to Installation Options for more information.

Note: Count of related To Do entries. There are some user interfaces that display a count of related To Do entries. This logic looks at foreign key characteristics linked to the To Do entry where the underlying object is a master or transaction object. It counts the distinct list of other non-closed To Do entries that reference the same characteristic(s).

Adjusting the To Do Role

When creating a To Do entry, for a To Do type, there may be attributes of the To Do entry that require that a special set of users work on it. This is sometimes referred to as 'rerouting the To Do entry' to a different role. This section highlights ways to override the role based on conditions specific to a To Do entry.

The information in this section only applies to automatic To Do Entries created by an algorithm or background process. It does not apply to manually created To Do Entries.

Refer to To Do Entries Reference A Role for the hierarchy used to determine the role .

Reroute Based on To Do's Message

As described in Suppress the Creation of the To Do Entry, To Do entries created for a given To Do type may have different messages based on the specific condition for a record. You may decide that based on the message, the To Do should be routed to a specific Role that is more knowledgeable about a specific error condition. The same message override configuration that may be used for suppression of a To Do entry, may alternatively be used to assign a specific Role. For example, if certain bill segment errors are always resolved by the rate specialist team, you can indicate the role for that team on the To Do type, message override.

Reroute Based on a Condition

Copyright © 2024, Oracle and/or its affiliates.

The To Do Pre-creation plug-ins may be used to assign a specific role based on some condition related to the To Do entry's specific data.

The product provides configuration on access group to define To Do assignment overrides. The override list defines a To Do type and To Do role. This is useful if your product associates access groups with specific master data for row level security purposes and you would also like to direct To Do entries related to this data to a specific role.

Oracle Utilities Customer Care and Billing also defines similar To Do assignment override configuration on CIS division and Account Management Group.

A pre-creation algorithm is responsible for checking if data linked to the To Do entry is related to a configuration object that supports an assignment override and then looks for an override for the To Do type. You can decide if this algorithm should be plugged in at the To Do Type level or the Installation level. The decision is based on how many To Do types you define overrides for. If it's a small number, then it makes more sense to plug the algorithm in for each To Do type that you define an override for. This ensure that the configuration is only checked for To Do entries that will have an override. However, if the number of To Do types that are eligible for overrides is larger or if the list changing or growing often, it may be better to plug the algorithm in at the Installation level to ensure that every To Do is checked for a possible override.

Note: Note that your product may already provide a base delivered pre-creation algorithm that is checking for this override. If so, that can be used by your implementation. If not, your implementation can provide an appropriate algorithm.

To Do Pre-creation Plug-ins

The system provides two To Do Pre-creation plug-in spots that allow for functionality to add or update information on the To Do Entry that is about to be created. The two plug-in spots are To Do type and installation options. In general, the installation option To Do pre-creation algorithms would be used for logic that applies to many To Do types and the To Do type pre-creation algorithms would be used for logic that is specific to that To Do type.

If at least one To Do pre-creation algorithm is plugged in on the To Do type, only those algorithms are executed when creating the To Do entry and the installation option algorithms are ignored. The system provides a base delivered To Do type pre-creation algorithm that executes the Installation option To Do pre-creation algorithms. This allows you to control whether the installation algorithms should also be executed for a To Do type that has specific algorithms and if so, when.

The following points highlight functionality supported by both To Do pre-creation plug-in spots.

- Indicate that the To Do entry should be suppressed (not created). The expectation is that this type of algorithm is configured on the To Do type rather than at the installation options level, as this allows for more granular conditions to be checked for a given To Do entry. Refer to Suppress the Creation of the To Do Entry for more information.
- Linking context specific data to the To Do entry using characteristics. Refer to Linking Related Information to a To Do Entry for more information.
- Overriding the Role of a To Do entry based on specific configuration related to the To Do's characteristic data. Refer to Adjusting the To Do Role for more information.

Assigning a Priority

As described in The Priority of a To Do Entry, a To Do type may reference a Calculate Priority algorithm. If so, this algorithm is called as part of the steps to create a To Do entry. The algorithm may use the drill key or any of the related information linked to the To Do as described above to determine priority information.

The Priority Of A To Do Entry

Some To Do entries may be more urgent to resolve than others. A To Do entry is associated with a priority level representing its relative processing order compared to other entries.

Priority level is assigned as follows:

Copyright © 2024, Oracle and/or its affiliates.

- If one or more **Calculate Priority** plug-ins are defined on the To Do entry's type, the system calls them to determine the entry's priority. They are called initially when a To Do entry is created and each time it gets updated. You may want to use this method if an entry's priority is based on context or time-based factors. For example, when priority takes into consideration account specific attributes. The system also provides a batch process (**F1-TDCLP**) that calls the calculate priority algorithms "at will" for non-closed To Do entries. This is useful when the priority should be reassessed periodically based on factors external to the To Do entry's information. Refer to To Do Type for more information on priority calculation algorithms. When the priority is determined by one of the algorithms, a log entry is created indicating that the priority was calculated.
- If a priority value has not been determined by a plug-in, the system defaults a To Do entry's initial priority to the value specified on its type.

A user may manually override a To Do entry's priority at any time. When a user overrides the priority, a log entry is created indicating that the priority was overridden. Notice that once a To Do entry's priority is overridden, **Calculate Priority** plug-ins are no longer called so as to not override the value explicitly set by the user.

Note: The system does not use priority values to control order of assignment nor processing of To Do entries. Priority is available to assist your organization with supporting a business practice that ensures higher priority issues are worked on first.

What Happens When You Work A To Do Entry

Many user interface tools allow the user to indicate that they will start working on a To Do entry. Refer to Working On A To Do for more details. When a user works on an entry, the user is transferred to the maintenance page associated with the entry (the To Do entry's drill key). The page to which the user is taken is configured on the To Do type.

It is also possible to configure the To Do type to launch a script when a user works on a To Do entry rather than taking the user to a specific page. The script can check conditions related to the To Do entry to determine a more appropriate page or could be configured to walk the user through the steps required to resolve the To Do entry.

A script is linked to a To Do type based on its message number using the To Do type's message overrides. Refer to Executing A Script When A To Do Is Selected for more information.

In certain pages, taking an action to work on a To Do entry automatically assigns the To Do entry to the user. For users in implementations that also take advantage of the assignment substatus values, the To Do entry is also updated to the **In Progress** substatus. Refer to The Lifecycle Of A To Do Entry for information about configuring the system to use the substatus values.

Monitoring A To Do Entry

A To Do type may reference a To Do monitor algorithm. The algorithms are executed by the To Do monitor background process (F1-TDMON). The process executes the monitor algorithms for any non-complete To Do entry whose To Do type references at least one monitor algorithm.

Monitor algorithms may be useful if there are business requirements where a To Do could be automatically completed based on criteria that the algorithm can review. For example, an algorithm may be used to detect if the situation that caused the To Do Entry to be generated has been remedied in the meantime, allowing for the To Do Entry to be completed.

To Do Entries Have Logs

Each To Do entry has a To Do log that maintains a record of the To Do's progress in the system. For example, the To Do log indicates when the To Do entry was created, when it was assigned to a user and to whom it was assigned, and when and by whom it was completed. Users can view the log to see who assigned them a particular To Do and whether any work has already been done on the To Do.

A log entry is created for all actions that can be taken on a To Do entry. Log entries are created for the following events:

Copyright © 2024, Oracle and/or its affiliates.

- A To Do entry is created (either by the system or by a user)
- A To Do entry is completed (either by the system or by a user)
- A user takes an open To Do entry
- A supervisor assigns a To Do entry
- The To Do entry's status is updated and the event is not already covered by another log entry type
- A user forwards an entry to another user or role
- A user sends back a To Do to the user who forwarded it
- A user manually adds a log entry to record details about the To Do's progress
- A user manually overrides the To Do entry's priority
- The To Do entry's priority was updated as a result of a calculate priority algorithm.

Fastpath: For information about the contents of log entries for each of the events, refer to Log Entry Events.

Implementing Additional To Do Entry Business Rules

If your business practice calls for additional validation rules or processing steps to take place after a To Do Entry is created or updated, you may want to take advantage of the **To Do Post Processing** plug-ins defined on To Do type.

For example, you may want to validate that To Do entries are only assigned to users with the proper skill levels needed to resolve them. Refer to **F1-VAL-SKILL** for a sample algorithm handling such validation.

To Do Entries May Be Routed Out Of The System

A To Do type can be configured so that its entries are interfaced to another system.

For example, a given To Do type can be configured to create an Email message whenever a new To Do entry is created. The following points describe how to do this:

- Define the name of the background process responsible for interfacing the new To Do entries to another system on the respective To Do type. The base package contains a batch process called **F1-TDEER** that can be used for most situations. This batch process invokes the **External Routing** algorithms defined on each entry's To Do type.
- Plug in an appropriate **External Routing** algorithm on the respective To Do type. The logic in this type of algorithm performs the interface efforts for a specific To Do entry. For example, if an Email message should be created for a To Do entry, the logic in the algorithm would compose and send the Email message(s) for a specific To Do entry.

To Do Information May Be Formatted By An Algorithm

A **To Do Information** algorithm may be plugged in on the installation record to format the standard To Do information that appears throughout the system. This algorithm may be further overridden by a corresponding plug-in on the To Do Type.

Periodically Purging To Do Entries

The To Do Entry is configured for Information Lifecycle Management. This is the recommended tool for managing older records.

A purging batch process is also supplied **F1-TDPG**. However, it has some limitations that should be considered. The purging process does not perform any referential integrity. As such, if you are proactively purging records that are not very old, they could be referenced by other objects that are still active.

The process purges **Completed** To Do entries. This background process offers you the following choices:

- You can purge all To Do entries older than a given number of days.
- You can purge To Do entries for a specific list of To Do types that are older than a given number of days.

Copyright © 2024, Oracle and/or its affiliates.

- You can purge all To Do entries except for a specific list of To Do types that are older than a given number of days.

Note: Different retention periods for different types of To Do entries. Keep in mind that the purge program allows you to retain different types of entries for different periods of time.

Setting Up To Do Options

The topics in this section describe how to set up To Do management options.

- Installation Options on page 382
- Messages on page 382
- Feature Configuration on page 382
- Defining To Do Roles on page 383
- Defining To Do Types on page 384
- Implementing The To Do Entries on page 387

Installation Options

There are a number of installation options that may be configured to govern various aspects of To Do processing.

- A **To Do Information** algorithm may be plugged in on the Installation record. Refer to To Do Entries My be Formatted By An Algorithm for more information.
- A **To Do Pre-creation** algorithm may be plugged in on the Installation record to set additional information for a To Do entry before it is created. Refer to To Do Pre-creation Plug-ins for more information.
- A **Next To Do Assignment** algorithm must be plugged into the Installation record if your organization opts to use the next assignment feature supported by the Current To Do Zone. The algorithm is responsible for determining the next To Do entry the user should work on. Make sure you provide users with security access rights to the zone's next assignment action. Refer to the Work Distribution zone for more information.
- If your To Do entries reference characteristics related to your global context data and your product supports sidebar alerts generated by algorithms, you may want configure an algorithm to display an alert if an entry is **Open** or **Assigned** for the data currently in context. Refer to your product’s documentation to determine if these types of alerts are supported.

Messages

You need only set up new messages if you use algorithms to create To Do entries or prefer different messages than those associated with the base package's To Do types.

Feature Configuration

The base package is provided with a generic **Activity Queue Management** Feature Configuration type. You may want to set up a feature configuration of this type to define any To Do management related options supporting business rules specific to your organization.

For example, the base package provides the following plug-ins to demonstrate a business practice where To Do entries are only assigned to users with the proper skill levels to work on them.

- The base **To Do Post Processing** To Do Type algorithm **F1-VAL-SKILL** validates that a user has the necessary skill levels required to work on a given To Do entry.
- The base **Next To Do Assignment** installation options algorithm **F1-NEXT-TDSK** has the ability to assign To Do entries to users that have the proper skills to work on them. This plug-in is applicable if your organization practices work distribution "on demand."

Copyright © 2024, Oracle and/or its affiliates.

You must set up such an **Activity Queue Management** feature configuration if you want to use any of the above base package plug-ins.

The following points describe the various **Option Types** provided with the base package:

- **Skill.** This option provides a reference to a skill category. For example, if you were using characteristics to represent skill categories then you should reference each characteristic type using this option.
- **Override Skill.** This option provides an override skill information reference for a specific message. For example, if you were using a To Do Type characteristic to specify an override skill category and level for a specific message category / number then you would reference this characteristic type using this option.

Note: Skill Setup. Refer to the description of the above base package algorithms for further details on how to setup skill level information.

Note: More Options. Your implementation may define additional options types. You do this by add new lookup values to the lookup field **F1QM_OPT_TYP_FLG**.

Note: Only one. The system expects only one **Activity Queue Management** feature configuration to be defined.

Defining To Do Roles

This section describes the control table used to maintain To Do roles.

- To Do Role - Main
- To Do Role - To Do Types

To Do Role - Main

The **Main** page is used to define basic information about a To Do role.

To maintain this information, select **Admin** > **General** > **To Do Role**.

Description of Page

Enter a unique **To Do Role** and **Description** for the To Do role.

The grid contains the ID of each **User** that may view and work on entries assigned to this role. The First Name and Last Name associated with the user is displayed adjacent.

Note: System Default Role. The system supplies a default role **F1_DFLT** linked to each system To Do type. This is done so that To Do functionality may be tested prior to the creation of appropriate business oriented To Do roles.

In addition, various "type" objects or algorithms may reference a To Do role to use when creating a To Do for a given business scenario. This is dependent on your specific product.

To Do Role - To Do Types

The **To Do Types** page defines the To Do types that may be viewed and worked on by users belonging to a given To Do role.

To maintain this information, select **Admin** > **General** > **To Do Role** and navigate to the **To Do Types** page.

Description of Page

Enter the ID of each **To Do Type** whose entries may be viewed and worked on by the role.

Use As Default is a display-only field that indicates if the role is assigned to newly created entries of this type. You may define the default role for a given To Do type on the To Do Type maintenance page.

CAUTION: If you remove a To Do type where this role is the default, you must define a new role as the default for the To Do type. You do this on the To Do Type maintenance page.

Copyright © 2024, Oracle and/or its affiliates.

Defining To Do Types

This section describes the control table used to maintain To Do types.

- To Do Type - Main on page 384
- To Do Type - Roles on page 384
- To Do Type - Sort Keys on page 385
- To Do Type - Drill Keys on page 385
- To Do Type - Message Overrides on page 386
- To Do Type - To Do Characteristics on page 386
- To Do Type - Algorithms on page 386

To Do Type - Main

The **Main** page is used to define basic information about a To Do type.

Fastpath: Refer to The Big Picture Of To Do Lists for more information about To Do types and To Do lists in general.

To maintain this information, select **Admin** > **General** > **To Do Type**.

CAUTION: Important! If you introduce a To Do type, carefully consider its naming convention. Refer to System Data Naming Convention for more information.

Description of Page

Enter a unique **To Do Type** and **Description** for the To Do type.

Owner indicates if this entry is owned by the base package or by your implementation (**Customer Modification**).

Use the **Detailed Description** to provide further details related to the To Do Type.

Enter the default **Priority** of To Do entries of this type in respect of other To Do types. Refer to The Priority Of A To Do Entry for more information.

For **To Do Type Usage**, select **Automatic** if To Dos of this type are created by the system (i.e., a background process or algorithm). Select **Manual** if a user can create a To Do of this type online.

Define the **Navigation Option** for the page into which the user is transferred when drilling down on a To Do entry of this type.

Use **Creation Process** to define the background process, if any, that is used to manage (i.e., create and perhaps complete) entries of this type. A **Creation Process** need only be specified for those To Do types whose entries are created by a background process. Refer to To Do Entries Created By Background Processes for more information.

Use **Routing Process** to define the background process that is used to download entries of a given type to an external system, if any. A **Routing Process** need only be specified for those To Do types whose entries are routed to an external system (e.g., an Email system or an auto-dialer). Refer to To Do Entries May Be Routed Out Of The System for more information.

Use **Message Category** and **Message Number** to define the message associated with this To Do type's entries. Note: this message will only be used if the process that creates the To Do entry does not supply a specific message number. For example, the process that creates To Do entries that highlight bill segments that are in error would not use this message; rather, the entries are marked with the message associated with the bill segment's error.

Use the characteristics collection to define a **Characteristic Type** and **Characteristic Value** common to all To Do entries of this type. You may enter more than one characteristic row for the same characteristic type, each associated with a unique **Sequence** number. If not specified, the system defaults it to the next sequence number for the characteristic type.

To Do Type - Roles

The **Roles** page defines the roles who may view and work on entries of a given To Do type.

Copyright © 2024, Oracle and/or its affiliates.

To maintain this information, select **Admin** > **To Do Type** > **Search** and navigate to the **Roles** page.

Description of Page

Enter each **To Do Role** that may view and work on entries of a given type. Turn on **Use as Default** if the role should be assigned to newly created entries of this type. Only one role may be defined as the default per To Do type.

Fastpath: Refer to To Do Entries Reference A Role for more information about roles and To Do entries.

To Do Type - Sort Keys

The **Sort Keys** page defines attributes that may be used to sort To Do entries.

The To Do Information zone on the To Do Management portal provides the ability to search for To Do entries using sort keys and includes up to five sort key values as columns in the results, which may be used for sorting as required.

The To Do list page includes an **Extra Information** option to allow To Do entries to be sorted by a selected sort key. Note the default sort key for this type of sorting is also defined on To Do type.

Note that this information is captured as a string and is meant for sorting purposes. The system does not capture data type information to be able to format the information for display. For example, dates are not going to display as per the user's display profile configuration.

To maintain this information, select **Admin** > **To Do Type** > **Search** and navigate to the **Sort Keys** page.

CAUTION: Do not change this information unless you are positive that the process / algorithm that creates entries of a given type stores this information on the entries.

Description of Page

The following fields display for each sort key.

Sequence is the unique ID of the sort key.

Description is the description of the sort key that appears on various pages.

Use as Default indicates the default sort key that is initially used when a user opens the To Do List page. Only one sort key may be defined as the default per To Do type.

Sort Order indicates whether the To Do entries should be sorted in **Ascending** or **Descending** order when this sort key is used on the To Do List page.

Owner indicates if this entry is owned by the base package or by your implementation (**Customer Modification**).

To Do Type - Drill Keys

The **Drill Keys** page defines the keys passed to the application service (defined on the Main page) when you drill down on an entry of a given type.

To maintain this information, select **Admin** > **To Do Type** > **Search** and navigate to the **Drill Keys** page.

CAUTION: Do not change this information unless you are positive that the process / algorithm that creates entries of a given type stores this information on the entries.

Description of Page

Navigation Option shows the page into which the user is transferred when drilling down on a To Do entry of this type.

The following fields display for each drill key.

Sequence is the unique ID of the drill key.

Table and **Field** are passed to the application service when you drill down on an entry of a given type.

Owner indicates if this entry is owned by the base package or by your implementation (**Customer Modification**).

Copyright © 2024, Oracle and/or its affiliates.

To Do Type - Message Overrides

The **Message Overrides** page is used if you want To Do entries that reference a given message category / number to be routed to a specific To Do role or suppressed altogether or if you want to associate a script to a given message category / number.

Fastpath: Refer to To Do Entries Reference A Role and Suppress the Creation of the To Do Entry and Adjusting the To Do Role for more information.

To maintain this information, select **Admin** > **To Do Type** > **Search** and navigate to the **Message Overrides** page.

Description of Page

The following fields display for each override.

Message Category and **Number** allow the message to be overridden.

Exclude To Do Entry indicates if a To Do entry of this type that references the adjacent **Message Category** and **Number** should not be created.

Override Role indicates the to do role to which a To Do entry of this type that references the adjacent **Message Category** and **Number** should be addressed. This field is protected if **Exclude To Do Entry** is on.

Script indicates the script that should execute when a user drills down on a To Do entry of this type that references the adjacent **Message Category** and **Number**. This field is protected if **Exclude To Do Entry** is on. Refer to What Happens When You Work A To Do Entry for more information.

To Do Type - To Do Characteristics

The **To Do Characteristics** page defines characteristics that can be defined for To Do entries of this type. The characteristic types for characteristics that are linked to the To Do entry as a result of a pre-creation algorithm do not need to be defined here.

To maintain this information, select **Admin** > **General** > **To Do Type** > **Search** and navigate to the **To Do Characteristics** page.

Turn on the **Required** switch if the **Characteristic Type** must be defined on To Do entries of this type.

Enter a **Characteristic Value** to use as the default for a given **Characteristic Type** when the **Default** switch is turned on. Use **Sequence** to control the order in which characteristics are defaulted.

To Do Type - Algorithms

The **To Do Algorithms** page defines the algorithms that should be executed for a given To Do type.

To maintain this information, select **Admin** > **To Do Type** > **Search** and navigate to the **Algorithms** page.

Description of Page

The grid contains **Algorithms** that control important To Do functions. If you haven't already done so, you must set up the appropriate algorithms in your system. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence Number** and **Algorithm** for each system event. You can set the **Sequence Number** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

The following table describes each **System Event**.

System Event	Optional / Required	Description
Calculate Priority	Optional	Algorithms of this type may be used to calculate a To Do entry's priority. Refer to The Priority of a To Do

Copyright © 2024, Oracle and/or its affiliates.

System Event	Optional / Required	Description
		Entry for more information on when this plug-in is called. Note that it is not the responsibility of the algorithms to actually update the To Do entry with the calculated priority value but rather only return the calculated value. The system carries out the update as necessary. If more than one algorithm is plugged-in the system calls them one by one until the first to return a calculated priority.
External Routing	Optional	Algorithms of this type may be used to route a To Do entry to an external system. The base package **F1-TDEER** background process invokes the algorithms for every To Do entry that its type references the process as the **Routing Process** and that the entry was not already routed. The background process marks an entry as routed by updating it with the batch control's current run number. If more than one algorithm is plugged-in the batch process calls them one by one until the first to indicate the To Do entry was routed.
To Do Information	Optional	We use the term "To Do information" to describe the basic information that appears throughout the system to describe a To Do entry. The data that appears in "To Do information" is constructed using this algorithm. Plug an algorithm into this spot to override the "To Do information" algorithm on installation options or the system default "To Do information" if no such algorithm is defined on installation options.
To Do Monitor	Optional	Algorithms of this type are executed by the To Do Monitor background process and may be used to periodically review a To Do entry and perform actions, if needed. Refer to Monitoring a To Do Entry for more information.
To Do Post-Processing	Optional	Algorithms of this type may be used to validate and/or further process a To Do entry that has been added or updated.
To Do Pre-Creation	Optional	Algorithms of this type are called when a To Do entry is being added. Refer to Refining a To Do Entry's Details at Creation Time for more information. To Do Type pre-creation algorithms override the Installation level pre-creation algorithms.

Implementing The To Do Entries

To enable the To Do entries visible in the To Do Type page, you must configure the system as follows:

Copyright © 2024, Oracle and/or its affiliates.

- Define the To Do roles associated with each To Do type and link the appropriate users to them. Once you have defined the roles appropriate for your organization's To Do types, remove the reference to this system default role **F1_DFLT**. Refer to To Do Entries Reference A Role for more information.
- For any To Do Type that is provided for a specific background process, the To Do simply needs to reference the appropriate Creation Background Process. When the background process is scheduled, To Dos are created based on the logic of the related background process. This applies to To Dos Created for Object-Specific Error Conditions and Dedicated To Do Background Processes.
- For any To Do Type that is provided for creation by an algorithm or other process, there may be configuration required to populate that To Do type as an algorithm parameter or as an attribute on a control table.

Note: Refer to the description of the To Do type for more information.

Chapter 7

Background Processes

Topics:

- Understanding Background Processes
- Defining Batch Controls
- Defining File Integration Types
- On-Line Batch Submission
- Tracking Batch Processes
- The Big Picture of Requests

This chapter covers various topics related to background processes. Besides providing an overview of background process functionality, the various tools available within the application to define, submit and monitor background processes are covered.

Note: Your specific source application may have additional background process topics. Please refer to the documentation section that applies to your source application for more information.

Understanding Background Processes

This section describes various topics related to the background processes that perform many important functions throughout your product such as:

- Processing To Do Entries
- Monitor processes that select records in a given state to progress them to their next state in their lifecycle
- Processes that purge data
- Processes that extract data
- And many more...

- Background Processing Overview on page 390
- Parallel Background Processes on page 391
- Parameters Supplied To Background Processes on page 392
- Processing Errors on page 398
- Error Post-Processing Logic on page 398
- Post-Processing Logic on page 398
- Timed Batch Processes on page 399
- Monitor Background Processes on page 399
- Plug-in Driven Background Processes on page 400
- How to Re-extract Information on page 413
- How to Submit Batch Jobs on page 414
- How to Track Batch Jobs on page 414
- How to Restart Failed Jobs and Processes on page 414
- Assessing Level of Service on page 414
- Batch Resource Groups on page 414

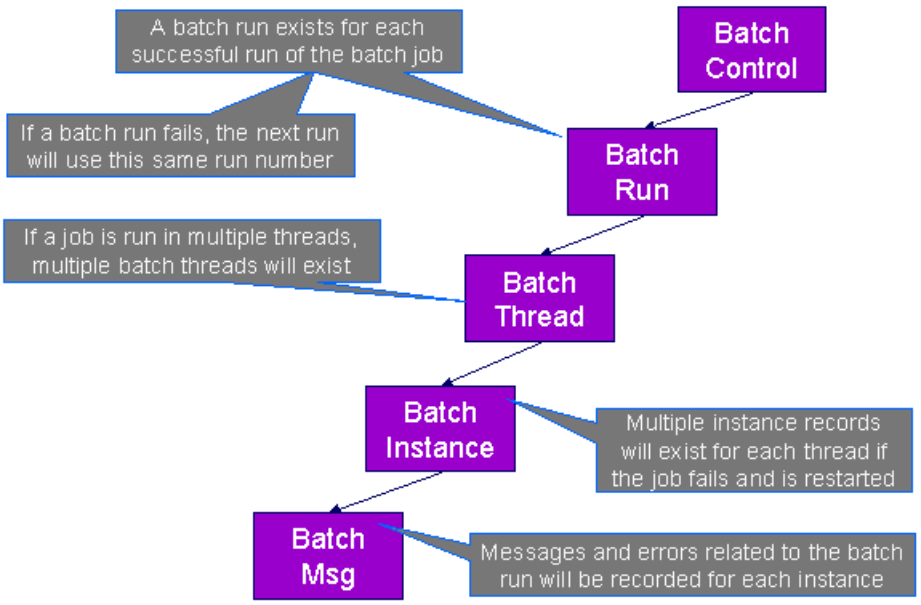
Background Processing Overview

While the system relies on a scheduler to secure and execute its background processes, there are additional issues that you should be familiar with:

- Batch control records are used for the following purposes:
  - Define the code that executes the logic associated with the background process.
  - For processes that extract information, the batch control record defines the next batch number to be assigned to new records that are eligible for extraction. For example, the batch control record associated with the process that routes To Do entries to an external system defines the next batch number to be assigned to new To Do entries that are configured with this batch control. When this To Do external routing process next runs, it selects all To Do entries marked with the current batch number (and increments the next batch number).
  - The batch control record for each background process organizes audit information about the historical execution of the background process. The system uses this information to control the restart of failed processes. You can use this information to view error messages associated with failed runs.
  - Many processes have been designed to run in parallel in order to speed execution. For example, the process that applies updates for a migration data set import for CMA can be executed so that multiple "threads" are processing a different subset of records (and multiple threads can execute at the same time). Batch control records associated with this type of process organize audit information about each thread in every execution. The system uses this information to control the restart of failed threads. Refer to Parallel Background Processes for more information.
  - Some processes define extra parameters. These parameters are defined with the batch control. Default values may also be captured for each parameter. They will be used when the background process is submitted on-line.

The following diagram illustrates the relationships that exist for batch control records.

Copyright © 2024, Oracle and/or its affiliates.



Results of each batch run can be viewed using the Batch Run Tree page.

Refer to Batch Scheduler Integration for information about the integration with the Oracle Scheduler.

Parallel Background Processes

Many processes have been designed to run in parallel in order to speed execution. This is referred to as running the process with multiple “threads”.

The system provides two strategies for distributing the data to the multiple threads.

- **Thread Level SQL Select**. This strategy is sometimes referred to as the “thread iterator” strategy. In this strategy, the batch job uses the primary key to figure out how to evenly distribute key ranges to each thread. Each thread is then responsible for selecting the records. In this strategy, the threads should also re-select the data periodically to release the cursor, which aids in performance. Note that this strategy is preferred but may only be used under the following conditions:
  - The data from only one maintenance object is being processed.
  - The primary key for the maintenance object is a single, numeric system generated key.

  Note: Parameters may be used to override the low and high id. Refer to Parameters Supplied to Background Processes for more information.
- **Job Level SQL Select**. This strategy is sometimes referred to as the “standard commit” strategy. In this strategy, the keys for the records to be processed by the batch job are all selected first and stored in a temporary table. The batch job then supplies each thread with a range of keys that it should process. This strategy is used if multiple maintenance objects are being processed by the batch job; if the primary key of the maintenance object has multiple parts or if the primary key is non-numeric.

The multi-threading logic relies on the fact that primary keys for master and transaction data are typically system generated random keys. In addition, if the data is partitioned, it is expected to be partitioned based on the primary key.

Note: The detailed description in the metadata for each batch control provided with the system should indicate if it may be run in parallel. Note that the strategy used is not typically indicated in the detailed description.

Copyright © 2024, Oracle and/or its affiliates.

> **Note: Overriding the thread ranges**. Your implementation has the ability to override the thread ranges if certain data in your system takes longer to process. For example, imagine you have a single account in Oracle Utilities Customer Care and Billing that has thousands of service agreements (maybe the account for a large corporation or a major city). You may want to set up the thread ranges to put this large account into its own thread and distribute the other accounts to the other threads. To do this, you should create the appropriate batch thread records ahead of time in a status of **Thread Ready** (**50**) with the key ranges pre-populated. Note that the base product does not provide the ability to add batch thread records online. If you are interested in more information about this technique, contact Customer Support.

- Optimal Thread Count on page 392

Optimal Thread Count

Running a background process in multiple threads is almost always faster than running it in a single thread. The trick is determining the number of threads that is optimal for each process.

> **Note:** A good rule of thumb is to have one thread for every 100 MHz of application server CPU available. For example if you have four 450 MHz processors available on your application server, you can start with 18 threads to begin your testing: (450 * 4) / 100 = 18.

This is a rule of thumb because each process is different and is dependent on the data in your database. Also, your hardware configuration (i.e., number of processors, speed of your disk drives, speed of the network between the database server and the application server) has an impact on the optimal number of threads. Please follow these guidelines to determine the optimal number of threads for each background process:

- Execute the background process using the number of threads dictated by the rule of thumb (described above). During this execution, monitor the utilization percentage of your application server, database server and network traffic.
- If you find that your database server has hit 100% utilization, but your application server hasn't one of the following is probably occurring:
  - There may be a problematic SQL statement executing during the process. You must capture a database trace to identify the problem SQL.
  - It is also possible that your commit frequency may be too large. Commit frequency is a parameter supplied to every background process. If it is too large, the database's hold queues can start swapping. Refer to Parameters Supplied to Background Processes for more information about this parameter.
- It is normal if you find that your application server has hit 100% utilization but your database server has not. This is normal because, in general, all processes are CPU bound and not IO bound. At this point, you should decrease the number of threads until just under 100% of the application server utilization is achieved. And this will be the optimal number of threads required for this background process.
- If you find that your application server has not hit 100% utilization, you should increase the number of threads until you achieve just under 100% utilization on the application server. And remember, the application server should achieve 100% utilization before the database server reaches 100% utilization. If this proves not to be true, something is probably wrong with an SQL statement and you must capture an SQL trace to determine the culprit.

Another way to achieve similar results is to start out with a small number of threads and increase the number of threads until you have maximized throughput. The definition of "throughput" may differ for each process but can be generalized as a simple count of the records processed in the batch run tree. For example, in the Billing background process in Oracle Utilities Customer Care and Billing, throughput is the number of bills processed per minute. If you opt to use this method, we recommend you graph a curve of throughput vs. number of threads. The graph should display a curve that is steep at first but then flattens as more threads are added. Eventually adding more threads will cause the throughput to decline. Through this type of analysis you can determine the optimum number of threads to execute for any given process.

Parameters Supplied To Background Processes

This section describes the various types of parameters that are supplied to background processes.

Copyright © 2024, Oracle and/or its affiliates.

General Parameters

The following information is passed to every background process.

- **Batch code.** Batch code is the unique identifier of the background process.
- **Batch thread number.** Thread number is only used for background processes that can be run in multiple parallel threads. It contains the relative thread number of the process. For example, if the billing process has been set up to run in 20 parallel threads, each of the 20 instances receives its relative thread number (1 through 20). Refer to Optimal Thread Count for Parallel Background Processes for more information.
- **Batch thread count.** Thread count is only used for background processes that can be run in multiple parallel threads. It contains the total number of parallel threads that have been scheduled. For example, if the billing process has been set up to run in 20 parallel threads, each of the 20 instances receives a thread count of 20. Refer to Optimal Thread Count for Parallel Background Processes for more information.
- **Batch rerun number.** Rerun number is only used for background processes that download information that belongs to given run number. It should only be supplied if you need to download an historical run (rather than the latest run).
- **Batch business date.** Business date is only used for background processes that use the current date in their processing. For example, a billing process may use the business date to determine which bill cycles should be downloaded. If this parameter is left blank, the system date is used. If supplied, this date must be in the format YYYY-MM-DD. Note: this parameter is only used during QA to test how processes behave over time.
- **Override maximum records between commits.** This parameter is optional and overrides each background process's Standard Commit. You would reduce this value, for example, if you were submitting a job during the day and you wanted more frequent commits to release held resources. You might want to increase this value when a background process is executed at night (or weekends) and you have a lot of memory on your servers.
- **Override maximum minutes between cursor re-initiation**. This parameter is optional and overrides each background process's Standard Cursor Re-Initiation Minutes. You would reduce these values, for example, if you were submitting a job during the day and you wanted more frequent commits to release held resources (or more frequent cursor initiations). You might want to increase these values when a background process is executed at night (or weekends) and you have a lot of memory on your servers.
- **User ID.** Please be aware of the following in respect of user ID:
  - Both the user submitting the job and the user ID recorded on the batch submission should have access to the application service for the batch control that secures execution.
  - Any batch process that stamps a user ID on a record it creates or updates uses this user ID in applicable processing.
  - This user ID's display profile controls how dates and currency values are formatted in messages.
- **Password.** Password is not currently used.
- **Language Code.** Language code is used to access language-specific control table values. For example, error messages are presented in this language code.
- **Trace Program Start**, **Trace Program Exit**, **Trace SQL** and **Trace Output**. These switches are only used during QA and benchmarking. If trace program start is checked, a message is displayed whenever a program is started. If trace program exit is checked, a message is displayed whenever a program is exited. If trace SQL is checked, a message is displayed whenever an SQL statement is executed. If trace output is checked, special messages formatted by the background process are written.

Note: The information displayed when the trace output switch is turned on depends on each background process. It is possible that a background process displays no special information for this switch.

Common Additional Parameters

Each batch control supports the definition of additional parameters. There are some additional parameters that are common to all batch processes or common to a specific type of batch process. The batch control should be delivered with the appropriate additional parameters. However, when new additional parameters are introduced, existing batch controls may not be updated with the new additional parameter.

The following table highlights the common parameters that may be linked to a batch control. Note that for batch parameters, although there is a sequence number that controls the displayed order of the parameter, the batch process

Copyright © 2024, Oracle and/or its affiliates.

does not use the sequence to identify a particular parameter but rather uses the parameter name. In some cases multiple parameter names are supported (a 'camel case' version and an 'all caps' version).

Parameter Name	Description	Additional Comments
MAX-ERRORS / maxErrors	Each of the batch processes has, as part of its run parameters, a preset constant that determines how many errors that batch process may encounter before it is required to abort the run. A user can override that constant using this parameter.	The input value must be an integer that is greater than or equal to zero. The maximum valid value for this parameter is 999,999,999,999,999.
DIST-THD-POOL	Each batch process executes in a thread pool. This parameter is only necessary if the batch process should execute in a different thread pool than the default thread pool.	The default thread pool name is **DEFAULT**.
emailMode	When the batch job is submitted with an associated email address, the default logic is to send an email when the job completes regardless of success or failure. Use this parameter to limit the email based on the status of the job when it ends.	Valid Values • **ERROR** - send an email only when the job ends in Error status. • **SUCCESS** - send an email only when the job ends in successfully. • **ALL** - always send an email only when the job ends. (This is the default.)
The following parameters are only applicable to jobs that use the **Thread Level SQL Select** method of distributing work to threads as described in Parallel Background Processes.		
overrideLowIdValue	Specifies a new low id to use in calculating the range for a thread. The framework by default assumes that the Id is between 0's (e.g. 000000000) and 9's (e.g. 9999999999), but this parameter will override the low value.	The parameter value can be an actual number or it can be set to **auto**. If **auto** is configured, it is set to the lowest current value on the database table associated with the background process.
overrideHighIdValue	Specifies a new high id to use in calculating the range for a thread. The framework by default assumes that the Id is between 0's (e.g. 000000000) and 9's (e.g. 9999999999), but this parameter will override the high value.	The parameter value can be an actual number or it can be set to **auto**. If **auto** is configured, it is set to the highest current value on the database table associated with the background process.
idRangeOverrideClass	Use this parameter to specify a custom class to do thread range calculation. During batch execution, this override class is instantiated and the setter methods called to initialize the Ids as required. The low and high getter methods are called to retrieve the high and low ids to be used for the run.	The class name specified must implement interface com.splwg.base.api.batch.BatchIdRangeOverride.
The following parameters are only applicable to jobs that perform a single commit, for example for extract batch jobs.		
numRecordsToFlush	This parameter defines how frequently to flush the Hibernate cache to prevent	

Copyright © 2024, Oracle and/or its affiliates.

Parameter Name	Description	Additional Comments
	high heap consumption and Out Of Memory Errors.	

Specific Batch Parameters

Some background processes define additional parameters that are specific to their functionality. When a process receives additional parameters, they are defined and documented in the batch control entry in the application.

- Indicating a File Path or File Name on page 395
- Encrypting and Signing Files on page 395
- Manifest File on page 396
- Sharing Customer Exported Data on page 397

Indicating a File Path or File Name

Some of the system background processes use extra parameters to indicate a File Path and/or File Name for an input file or an output file. For example, most extract processes use File Path and File Name parameter to indicate where to place the output file.

For the file path parameter, refer to Referencing URIs for several points related to populating the file path and the supported syntax.

For the file name parameter, there are extract processes that support defining a file name that ends in ".zip" or ".gz", which will automatically result in writing the file with zip or gzip compression (based on the suffix). Similarly, there are upload processes that support decompression of files using the appropriate method based on the file name ending in ".zip" or ".gz". If the extract or upload process supports compression / decompression, it should be indicated accordingly in the description of the File Name parameter. If a batch control doesn't explicitly indicate that it supports compression or decompression in its description, the assumption should be that it is not supported.

In addition, there are extract and import processes that support automatic signing, encryption and decryption of the file content using PGP standards. File names ending in .pgp will be parsed to determine whether encryption, signing, compression or a combination of the three applies. If the file has a .pgp extension and requires decryption or signature verification, a runtime error will be issued if no encryption or signing key ring parameters are supplied.

Encrypting and Signing Files

Some files created by data export related batch processes, may need to be encrypted and/or digitally signed for security purposes. Similarly, batch upload processes may need to verify the signature and/or decrypt the content of the file being imported.

New batch parameters control whether encryption or digital signing applies to files extracted or uploaded by the process. Both encryption and signing use Pretty Good Privacy (PGP) standards.

Processes that include a file name parameter, such as those using the plug-in driven extract template, support a file name extension of '.pgp'. Files ending with this extension are encrypted and/or signed using the encryption key parameters described in the next section.

Processes that have an internally constructed format for the file name, such as those using the generalized initial export template, perform encryption and/or signing according to whether the encryption parameters are supplied. An extension of '.pgp' is added to the constructed file name.

Extract Process Encryption Parameters

Parameter Name	Description	Additional Comments
encryptExternalKey	Use this key ring parameter to configure the external key provided by the recipient of the file.	If populated, the batch output file will be encrypted using a runtime generated key. The generated key will be encrypted using the active public key stored on the key ring. The

Copyright © 2024, Oracle and/or its affiliates.

Parameter Name	Description	Additional Comments
		recipient will use their private key to decrypt the generated key so the file may be decrypted. Only external public key rings may be used.
encryptSignatureKey	Use this key ring parameter to configure the internal key used to sign the file.	If populated, the batch output file will be signed using the active private key stored on the key ring. The assumption is that the corresponding public key has been shared with the recipient. Only internal file encrypting / signing key rings may be used.

Upload Process Encryption Parameters

Parameter Name	Description	Additional Comments
decryptInternalKey	Use this key ring parameter to configure the internal key to be used to decrypt the file to be uploaded.	If populated, the generated key received with the input file will be decrypted using the active private key stored on the key ring. The data is then decrypted using the generated key. Only internal file encrypting / signing key rings may be used.
decryptSignatureKey	Use this key ring parameter to configure the public signature key provided by the sender of the file.	If populated, the input file signature will be verified using the active public key stored on the key ring. Only external key rings may be used.

You may use the Batch Control Query portal to find all batch controls supporting these options.

Refer to File Encryption and Signing Keys for more information on maintaining internal and external encryption keys.

Manifest File

Batch processes that create files, for example data export related batch processes, may create one or more files in each processing thread. External applications that consume these files need to process them only when the entire batch process is successfully completed. To automate the consumption of such files, the external application may leverage a function to be triggered upon the creation of a manifest file to process all the files referenced by the manifest. The manifest file includes details such as content hash value, file size information etc that assist in confirming that file content is completed and has not been tampered with prior to processing the files.

The following sections provide additional information about manifest files.

The Manifest Option Batch Parameter

Manifest file creation is supported on certain base product extract related batch controls only. These batch controls include the **Manifest Option** batch parameter that controls whether a manifest file should be created or not upon successful completion of the batch process. You may use the Batch Control Query portal to find all batch controls supporting this option.

Manifest File Information

Manifest file content is provided in JSON format and includes the following information:

- The version of the manifest details structure.
- The batch control code, run number and rerun number identifying the batch run producing the files
- Total number of files created by the batch run.
- Total number of records processed by the batch run.

Copyright © 2024, Oracle and/or its affiliates.

- The manifest creation timestamp in ISO format.
- The previously completed batch run number, if any.
- The manifest file produced by the previously completed batch run, if any. This may be used to ensure files are processed in a proper order and assist in detecting potential situations of missing files.
- A list of files created by the batch run that includes the following details for each file: the file's name, its size in bytes and the SHA-256 hash value of the file's content.

The manifest file name is constructed as follows:

```
[batch control]_[batch run number]_[batch rerun number]_manifest.json
```

The following is an example of the manifest file structure.

```
{
  -"version": 1,
  -"batchCode": -"HD-PDE",
  -"batchRunNumber": 106,
  -"batchRerunNumber": 0,
  -"totalFiles": 3,
  -"recordsProcessed": 10,
  -"timeStamp": -"2020-05-02T16:55:43.000353Z",
  -"previousBatchRunNumber":"105"
  -"previousManifestFileName":"HD-PDE_105_0_manifest.json",
  -"fileDetails": [
    {
      -"fileName": -"SOME_FILE_NAME_001.json.gz",
      -"fileSize": 1543,
      -"hashAlgorithm": -"SHA-256",
      -"hashValue": -"ff7d4e466f93cadc4a18ab009080916baff631e03b728e2f2684c4a4f92ccbe1"
    -},
    {
      -"fileName": -"SOME_FILE_NAME_002.json.gz",
      -"fileSize": 1182,
      -"hashAlgorithm": -"SHA-256",
      -"hashValue": -"78869e7ce971f28a70f5b35693a05726ba66109fecf492edb60291a62ae1ba9f"
    -},
    {
      -"fileName": -"SOME_FILE_NAME_003.json.gz",
      -"fileSize": 1182,
      -"hashAlgorithm": -"SHA-256",
      -"hashValue": -"78869e7ce971f28a70f5b35693a05726ba66109fecf492edb60291a62ae1ba9f"
    -}
  -]
}{
```

Verifying The Manifest File

The **Verify Manifest File** web service may be used by an external system to validate that the manifest file it has received for a specific batch run is not corrupted or tampered with. The service returns the manifest file name and the hash value of its content for the requested batch run allowing the caller to compare their file details with the returned values.

Sharing Customer Exported Data

Some data export related batch processes support an option that streamlines the process by which customer extracted data may be routed to and shared with Oracle so it can be uploaded to Oracle's data lake for analytic purposes. These batch controls include the **Copy to Oracle** batch parameter that controls whether a copy of the exported file should be sent to Oracle's data lake. You may use the Batch Control Query portal to find all batch controls supporting this option.

Note: The batch parameter is set to not share any content by default. Customers may opt in to share their data with Oracle by submitting such a batch process with the parameter explicitly set to do so.

Note: This option is only available in a cloud installation.

Copyright © 2024, Oracle and/or its affiliates.

Processing Errors

When a background process detects an error, the error may or may not be related to a specific object that is being processed. For example, if the program finds an error during batch parameter validation, this error is not object-specific. However, if the program finds an error while processing a specific bill, this error is object-specific. The system reports errors in one of the following ways:

- Errors that are not object-specific are written to the error message log in the Batch Run Tree.
- Some batch processes create entries in an "exception table" for certain object-specific errors. For example, an error detected in the creation of a bill in Oracle Utilities Customer Care and Billing may be written to the bill exception table. If an error is written to an exception table, it does not appear in the batch run tree. For each exception table, there is an associated to do entry process that creates a To Do Entry for each error to allow a user to correct the problem on-line.
- For some background processes, errors that do not result in the creation of an exception record may instead generate a To Do entry directly. For these processes, if you wish the system to directly create a To Do entry, you must configure the To Do type appropriately. Refer to To Do entry for object-specific errors for information about configuring the To Do type. If the background process detects an object specific error and you have configured the system to create a To Do entry, the error is not written to the batch run tree. If you have configured your To Do type to not create To Do entries for certain errors, these errors are written to the batch run tree.

Note: Some processes create exceptions and To Do entries. It is possible for a background process to create entries in an exception table and create To Do entries directly, depending on the error. Consider batch billing in Oracle Utilities Customer Care and Billing; any conditions that cause a bill or bill segment to be created in **error** status result in a record added to the bill exception table or the bill segment exception table. However, any object-specific error that is not related to a specific bill or bill segment or any error that prevents a bill or bill segment from being created may result in a To Do entry for the object-specific error.

Error Post-Processing Logic

The product supports executing one or more algorithms when a batch process encounters an error that causes execution to stop. This allows for some special processing to occur to handle the failure of the batch job. Algorithms for this plug-in spot receive the batch control, batch run number, batch processing business date, number of threads and the list of the ad hoc parameters of the batch job.

Note: This plug-in spot is available for all Java based batch programs. For programs of type **Java (converted)**, the individual batch programs need to explicitly support this plug-in spot. Unless otherwise noted, assume that a **Java (converted)** program does not support it.

The following are some examples of functionality that may be executed when a batch job fails:

- Another object or record that is monitoring the batch job may have its status updated to reflect the batch status.
- An outbound message service may be invoked to perform a task related to the failure.

Note that the units of work for all threads are committed prior to executing the error post-processing logic.

Post-Processing Logic

The product supports executing one or more algorithms after all the threads of a given batch job have completed. This allows for some special processing to occur at the end of a batch job. Algorithms for this plug-in spot receive the batch control, batch run number, batch processing business date, number of threads and the list of the ad hoc parameters of the batch job.

Copyright © 2024, Oracle and/or its affiliates.

> **Note:** This plug-in spot is available for all Java based batch programs. For programs of type **Java (converted)**, the individual batch programs need to explicitly support this plug-in spot. Unless otherwise noted, assume that a **Java (converted)** program does not support it.

The following are some examples of functionality that may be executed at the end of a batch job:

- Another dependent batch job can be kicked off. Note that this use case is only needed when the multiple dependent jobs are not part of a scheduler (which can also detect the successful end of one batch job so as to submit the next job).
- Statistics for the batch run may be analyzed and based on results, a To Do Entry may be sent to an administrator.
- If the current batch job is processing a large number of child records in multiple threads, a parent record could be updated to a different status or with some other audit information.

Note that the units of work for all threads are committed prior to executing the post-processing logic. The algorithm should perform standard error handling. If an error occurs in one of the post-processing algorithms, the overall batch job's status is set to Error so that it can be re-submitted to retry the logic in the finalize step.

Timed Batch Processes

Most batch jobs are submitted via a batch scheduler. In the absence of a scheduler, a batch control may be configured as "timed" triggering the framework to monitor and schedule these batch jobs as defined by the timer interval. The timer interval defines the desired interval between starts (in seconds). The system schedules new batch runs at each interval if the last instance of the job has completed.

When configuring a batch control as "timed", other default information must be provided, including the User ID and Language to use for submitting the job and the email address for notification, if desired.

Timed batch controls also include an Active setting, allowing for an implementation to temporarily stop further executions of the batch job (but retain the other timer settings).

Timed jobs are controlled by the default threadpool and not by a scheduler. When the **DEFAULT** threadpoolworker starts it will start executing any job for a Batch Control configured as **Timed** with the Timer Active set to **Yes**. This is whether the batch daemon or batch server is enabled or not.

Monitor Background Processes

In many areas of the system, functionality is driven from business object configuration as a BO driven record progresses through its lifecycle. Refer to Business Object Lifecycle for details. As part of that functionality, it is possible that a background process, called a monitor batch process, is used to execute functionality for the record. A single program is provided for the BO monitor functionality. The product also provides a batch control template (**F1-MNTPL**) that should be used as a starting point. Parameters are used to limit the records processed by maintenance object and other optional parameters that may further limit the records. The product typically provides at least one monitor batch control for each maintenance object that supports a configurable lifecycle on its business object.

This topic highlights the parameters supported by the monitor batch job. Not all parameters are applicable to all maintenance objects and therefore may not be configured on a given base monitor batch control.

Parameter Name	Description	Comments
maintenanceObject	Maintenance Object	For most base delivered batch controls, this parameter is delivered already populated with the value of the maintenance object value. Note that it is supported to leave this value blank, at which point, the program will determine the maintenance object (objects) to process by looking for an MO that refers to this batch control record as an option.
isRestrictedByBatchCode	Restrict by Batch Code	Set this to **true** to indicate whether the process should only select records that explicitly refer to this batch control on its current BO state. This is also referred to as "deferred" mode. If set to **false**, the

Copyright © 2024, Oracle and/or its affiliates.

Parameter Name	Description	Comments
		program includes all records that refer to the current batch control in its BO state and records that don't refer to any batch control in its current state (but monitor algorithms exist in the current state). This is commonly referred to as "periodic" mode. Note that if the value is not set at all, the program will determine whether to run it as "deferred" or "periodic" based on whether the batch code is configured on the MO option as a **State Monitor Process** ("deferred") or a **Periodic Monitor Process**.
restrictToType	Restrict by Related Type	This parameter is only applicable to maintenance objects that have a related 'type' object and the maintenance object has configured an option indicating the field for the related type column. This parameter may be used to limit the processing to records that are in the indicated type.
restrictToBusinessObject	Restrict by Business Object	This parameter may be used to limit the processing to records that are in the indicated business object.
restrictToBOStatus	Restrict by Status	This parameter may be used to limit the processing to records that are in the indicated status.
restrictToDate	Restrict By Date	Enter a valid date on the record's primary table to limit processing to records that are have a value in this field that is on or before the batch business date. Optionally, enter "+" or "-" followed by a number to shift the comparison date to the batch business date plus or minus the given number. For example, entering REVIEW_DT -3 will retrieve all records whose review date is on or before the batch business date minus 3 days.
sampleRecordNumber	Sample Record Number	This is not a commonly used parameter. It is only applicable when the monitor is used for a business use case that supports processing a subset of the records during a testing phase. For example, if the process is validating a large number of records, it may be an option to only validate every 100 records to determine if there are repeated validation errors that may indicate a common problem that may be solved to fix many errors.

Also note that when submitting a monitor process with multiple parallel threads, the program will use a **Thread Level SQL Select** strategy unless any of the following are true (in which case it will use the **Job Level SQL Select** strategy:

- The input maintenance object is left blank and the program finds more than one maintenance object that refers to this batch control in its options.
- A single MO is applicable but it has a multi-part primary key.
- A single MO is applicable and it has a single primary key, but it is a user defined key instead of a system generated key.
- The sample record number parameter is populated.

Plug-in Driven Background Processes

Although the product is delivered with a rich library of background processes, implementations may have business requirements that require new processes to be introduced. It is possible for an implementation to write a background process from scratch using a base process as a template. However, the product also provides base background processes that call algorithms to do the work that is needed. These are called plug-in driven background processes. There are three major types of plug-in driven batch processes:

Copyright © 2024, Oracle and/or its affiliates.

- Processes that act on records that are stored in the database in the system. These types of processes require SQL to select the records along with the logic to process the records.
- Processes that extract information to a file. These types of processes require SQL to select the records along with the logic to select the information and return it such that the process can write the information based on various settings.
- Processes that import data from a file and store new records in the system as a result. These types of processes require an algorithm that is able to map the data from the file to appropriate new records in the system

The subsequent sections provide more detail about the three types of plug-in driven background processes.

Ad-hoc Processes

A process that retrieves records in the system and does some action on them requires an algorithm to select the records to be processed and another algorithm to process the records. We refer to this type of process as an Ad-hoc Process. The base batch control Plug-in Driven Generic Template (**F1-PDBG**) may be used as a template.

The base processes implement standard background process functionality including parallel background process logic and the ability to create To Do entries for errors. This allows for an implementation to take advantage of the pre-built support and provide plug-ins that include the logic that is unique to the specific use case.

The following sections provide more information about the functionality for ad-hoc processes. Note that extract processes share some of the same functionality. Refer to Extract Record Processing for additional information specific to extract processes.

Select Records Algorithm

The first important algorithm to design when implementing a plug-in driven batch process is the Select Records algorithm, plugged in on the batch control page. This algorithm type must define the first parameter as the SQL. The batch job will directly access the SQL parameter value in the metadata (rather than invoking the algorithm). All other parameters are available for the algorithm to use for its own logic.

In addition, when invoking the algorithm, it must return the strategy to use (**Thread Level SQL Select** or **Job Level SQL Select**. Refer to parallel background processes for more information about the two strategies and when to use each. When choosing the **Thread Level SQL Select** strategy, the algorithm should return the name of the primary key in the Key Field parameter. In addition, the SQL should include a **BETWEEN** clause that includes the bind variables for the low and high ID for the ranges. See below for the bind variable syntax.

If the SQL statement includes variables that are determined at execution time, it must use bind parameters. Bind parameters are referenced in the SQL statement using a colon and a parameter name (for example **:parameter**). There are some variables provided by the system that are populated by the batch job at execution time. These have **f1_** as its prefix.

The system supports the following pre-defined bind parameters:

- **:f1_lowID** and **:f1_highID** - these should be used in the **BETWEEN** clause for the **Thread Level SQL Select** strategy. They should be selecting data from the Key Field that is returned as part of the Select Records algorithm. The batch job will substitute the appropriate ID range as required.

  Note: The table where this key field is selected from should be the first table in the FROM clause.
- **:f1_batchCode** and **:f1_batchNumber** - these are common attributes of the batch control that are referenced on a record for selection purposes. Note that the batch run number is set according to whether the batch job is a re-run of a previous run or not.
- **:f1_businessDate** - the batch job will populate the input batch business date, if populated otherwise the current date.

Note: The system supports both "f1." and "f1_" as a prefix for the bind variables. However, the "f1." prefix will result in an error if the SQL security property setting is turned on. As such, the underscore syntax is recommended. The period syntax remains for backwards compatibility.

For any other custom parameters, the Select Records algorithm may return one or more sets of field name / variable name / value where the variable name matches a bind variable in the SQL. The field name provides information about the length that assists the SQL binding logic to properly substitute the values. Note that the variable name cannot start with

Copyright © 2024, Oracle and/or its affiliates.

f1. as its prefix. The batch job will use the value returned by the algorithm to set the bind parameter in the SQL statement. Also note that the logic binds all variables as a string, so for date variables the SQL must include the "TO_DATE" function.

The plug in spot receives a list of the ad hoc parameters for the batch job as name / value pairs. If the list includes parameters whose values are to be used in selecting records, your algorithm may be used to identify the relevant batch parameter passed as input and populate the field name and output bind variable appropriately.

The product provides a base algorithm type for this plug-in spot that simply defines a parameter for the SQL. It also includes parameters for the strategy and the key field name. This algorithm type may be used by any custom batch process where the SQL does not rely on any special bind variables that must be determined. Simply create an algorithm for the algorithm type and provide the appropriate SQL. Refer to the algorithm type Select Records by Predefined Query (**F1-PDB-SR**) for more information.

Process Records Algorithm

The other important algorithm to design when implementing a plug-in driven batch process is the Process Record algorithm, plugged in on the batch control page. This algorithm is called for each record selected for the process. It receives all the information that was selected from the Select Records plug-in.

For the ad-hoc processing batch process, algorithms plugged into this spot are responsible for doing the work for each record based on the desired logic.

For the extract batch process, algorithms plugged into this spot are responsible for returning the data that should be written to the file. Many of the parameters for this plug-in spot are only applicable to extract processes. Refer to Extract Record Processing for more information on the functionality specific to extract processes.

Also note that algorithms for this plug-in spot will be passed two Booleans, isFirst and isLast, to indicate if the current work unit is the first and/or last for that thread. This allows for the plug-in to do additional work if needed. Note that the isFirst indication is available for both types of batch processes, ad-hoc and extract. However, the isLast indication is only applicable for the file extract batch. For the ad-hoc batch process this value will always be set to **false**. Extracts will always execute in a single database transaction. In a single transaction run, any error causes the run to be aborted so that it restarts from the beginning when resubmitted. This is done to avoid partial files being written along with inaccurate setting of the isLast element.

Configuring a New Process

The following points summarize the steps needed to implement a new background process that acts on records in the system using plug-ins for the specific functionality:

- Verify the SQL that the background process should execute. Keep in mind that all the data selected in the SQL is available to pass to the plug-in that processes the records. If the performance of the background process is important, be sure to consult with a DBA when designing the SQL.
  - If the SQL does not require any custom variables to substitute at runtime, create an algorithm for the base algorithm type **F1-PDB-SR** and configure the SQL. In addition, configure the strategy and the primary key name (for the **Thread Level SQL Select** strategy).
  - If the SQL requires custom variables, a new plug-in script must be designed and coded to populate the variable names and values using the algorithm entity **Batch Control - Select Records**. Besides defining the variables, the algorithm must also indicate the strategy and the primary key name (for the **Thread Level SQL Select** strategy). Define the algorithm type for the newly created script. The first parameter of the algorithm type must be the SQL as illustrated in the base algorithm type. Note that the other parameters are available for use by this algorithm type if needed. Define the algorithm, populating the SQL as appropriate (using the custom variables).
- Design the logic required to process each record. If your edge product does not deliver a suitable plug-in, create a plug-in script where the algorithm entity is **Batch Control - Process Record**. Note that the plug-in receives all the information selected in the SQL defined in the Select Records plug-in. For an ad-hoc process, the algorithm should perform whatever process is required based on the business use case. Note that the background process is responsible for committing the records.

Copyright © 2024, Oracle and/or its affiliates.

In either case, if a new processing script is required, define the algorithm type and algorithm for the newly created script.

- Create a batch control by duplicating the base template **F1-PDBG**. Plug in the algorithms created above and configure the parameters as appropriate. Note that you may configure custom ad hoc parameters on the batch control if required. Both base and custom batch parameter values are available to the Select Records and Process Records plug in algorithms.

Extract Record Processing

A process that selects records and produces output for each record to write to a file to integrate with external systems is called an Extract Process. The base batch control Plug-in Driven Extract Template (**F1-PDBEX**) may be used as a template.

Fastpath: Please read the topic Ad-hoc Processes. Its information about the Select Records plug-in and some of the information about the Process Record plug-in apply to extract processes. This topic focuses on aspects that are unique to extract processes.

The system provided process includes parameters to configure the file path and file name for the created file. The file path supports all the functionality described in the topic Referencing URIs. The file name supports system substitution variables, such as run number, thread number, user and date / time. The process also supports compression if an appropriate extension is used. Refer to the File Name parameter description in the batch control for more information.

Note: Refer to Flexible File Name / Writing Multiple Files for information about support for substituting a business value into the file name.

For extract processes, the Process Record algorithm is responsible for returning the data that should be written to the file in one or more XML instances along with the schema name(s) that describes the XML instance(s). The program will write the data to the file as per the format indicated in the File Format batch parameter. By default the service uses the OUAF format for date and time. To override this and use XSD format, configure the Date Time Format batch parameter to 'XSD".

If an existing schema satisfies the output requirements, it may be used. Otherwise, you should define a data area to indicate the output format of the records as appropriate. Note that you should define an appropriate Field for each element in the schema. This is needed for the fixed position File Format, but is good practice for any output format. Refer to **F1-PDBGenProcExtractRecord** for an example of a data area with fields defined for each schema element.

There are two options for designing and configuring the Process Record algorithm:

- Create a specific **Process Record** algorithm type that encapsulates the extract logic. The product provides a base algorithm type that illustrates the basic technique to follow. Refer to the algorithm type General Process - Sample Process Record Extract (**F1-GENPROCEX**) for more information.
- Create a file integration type that defines the records to be included in the extract and configure a plug-in driven batch control that references that integration type. Refer to Extract Using File Integration for more information about implementing this type of functionality.

Your specific edge product may provide other Process Record algorithm types out of the box. Use the algorithm type query and search for records for this algorithm entity.

Grouping by Record XML Node

When configuring an extract process that should produce the output in XML format, your Process Record plug-in may return multiple schemas with information that is all part of the same overall record. For example, the output could include account information and related service agreement details:

```
<account type="group"
    <accountId>1234567890</accountId>
    -...
</account>
<sa type="group"
    <saId>123457665</saId>
```

Copyright © 2024, Oracle and/or its affiliates.

```
    -...
</sa>
<sa type="group"
    <saId>1234588913</saId>
    -...
</sa>
```

In this case, it may be desired to wrap all that information in a grouping XML tag so that all the information for one record is grouped together. Because the Process Record plug-in could be returning header or footer records that are not part of a given selected record's extract information, the batch process doesn't know what schemas returned by the Process Record plug-in belong together logically. The output parameter 'record XML node' in the schema collection should be used to indicate the outer XML node to use to group related information together. For example:

```
<SchemaInstance>
<recordXMLNode>record</recordXMLNode>
<schemaName>CM-AccountRecord</schemaName>
<schemaType>F1DA</schemaType>
<data><account><accountId>1234567890</accountId>...</data>
</SchemaInstance>
<SchemaInstance>
<recordXMLNode>record</recordXMLNode>
<schemaName>CM-SAInfo</schemaName>
<schemaType>F1DA</schemaType>
<data><sa><saId>123457665</saId>... </data>
</SchemaInstance>
<SchemaInstance>
<recordXMLNode>record</recordXMLNode>
<schemaName>CM-SAInfo</schemaName>
<schemaType>F1DA</schemaType>
<data><sa><saId>1234588913</saId>... </data>
</SchemaInstance>
```

All schemas returned from a single call to the Process Record plug-in that have the same record XML node will be grouped in the written output within that XML tag, as per the example below:

```
<root>
<record>
<account>
    <accountId>1234567890</accountId>
    -...
</account>
<sa>
    <saId>123457665</saId>
    -...
</sa>
<sa>
    <saId>1234588913</saId>
    -...
</sa>
</record>
<record>
<account>
    <accountId>987654320</accountId>
    -...
</account>
<sa>
    -...
</sa>
</record>
</root>
```

Multi-threaded Extract

When extracting data, it's possible that the volume of data warrants running the job multi-threaded. In this case, a separate file is produced with the thread number included in the file name. You may proactively include the thread number as a substitution variable when indicating the file name. If not, the system appends the thread number.

Copyright © 2024, Oracle and/or its affiliates.

As a convenience, the system supports concatenating the extract files produced by the various threads at the end. The system does this by finding files whose file names match except for the thread number. As such, the file name may not contain date or time if the concatenation parameter is true. Please note the following with respect to the concatenation feature:

- The content of each file is concatenated together "as is". If the individual files have header, footer or summary information, there is no logic to consolidate that information in the concatenated file. If you want a single header, footer or summary, you must run the extract single threaded.
- Concatenation is not supported if the file name indicates the zip compression should be used.
- The individual thread files are retained.
- Note that if the file format is XML, the files written for the individual threads will be written with an extension of ".tmp".

Skipping Records

By default, the extract process expects one or more schemas to be returned by each call to the Process Record plug-in. (Ideally when designing any process, the Select Records algorithm only selects records that should be processed and the Process Record algorithm does not need to confirm whether the record should be part of the processing or not.) If however there is a need to check a condition in the Process Record plug-in and it is found that no data needs to be extracted for this record, the process record algorithm should return the 'is skipped' output parameter to **true** so that the program doesn't write an empty row.

Note that the base process records algorithm for file integration types **(F1-FILEX-PR)** will set the ‘is skipped’ parameter to **true** if the file integration record extract record algorithm returns no data.

Configuring a New Process

The points documented in the topic related to configuring a new adhoc process with respect to the Select Records algorithm apply to extract processes.

For the Process Records algorithm, it is responsible for returning one or more schema instances populated with information that should be written to the file. If your edge product does not deliver a suitable plug-in, create a plug-in script where the algorithm entity is **Batch Control - Process Record**. Note that the plug-in receives all the information selected in the SQL defined in the Select Records plug-in.

Refer to Configuring an Extract Process Using File Integration Type for information on designing extracts using this configuration.

If a new processing script is required, define the algorithm type and algorithm for the newly created script. Create a batch control by duplicating the base template **F1-PDBEX**. Plug in the algorithms created above and configure the parameters as appropriate. Note that you may configure custom ad hoc parameters on the batch control if required. Both base and custom batch parameter values are available to the Select Records and Process Records plug in algorithms.

Flexible File Name / Writing Multiple Files

By default, the data is written to the File Name defined in the batch parameter. The file name supports several dynamic substitution variables as defined in the detailed description for the batch parameter in the Plug-in Driven Extract Template batch control (**F1-PDBEX**). Also refer to Extract Record Processing for information about the file name and multi-threaded processes.

The system also supports more advanced file creation for a single batch run. Consider the following use cases

- An extract is for data associated with multiple service providers where each service provider should receive its own file. In this use case, the unit of work is the service provider so a file is produced for each work unit.
- An implementation operates with multiple jurisdictions represented by CIS Division. A certain extract should result in a file per division so that each division only gets its relevant data.
- An extract of Person information should separate individual person data from business person data. In addition, for each person, the contact information should result in a separate file. In this use case, one unit of work includes data to be written to more than one file.

To support the above use cases, the Process Record algorithm supports returning the File Name in the output:

Copyright © 2024, Oracle and/or its affiliates.

- The file name is included in the output Schema Instance group. You would populate this if you want the data for a specific Schema entry to go to this file name. In the person extract example above, the schema that includes the contact information would indicate the specific contact file.
- The file name is an output field outside the schema list. You would populate this if you don't have different files for different schemas but you still want to override the file name defined in the batch parameter. The service provider and division examples above would populate this parameter with the desired file name. The person extract example could populate this field for all the non-contact information.

The batch process checks the output after each call to the Process Records algorithm. For each schema, if there is a specific File Name associated with the schema, the data for that schema is written to that file. Otherwise, if the output file name is provided, the data is written to that file. Otherwise, the File Name in the batch parameter is used.

Note: Substitution Variables. After the call to Process Records, if there are any system variables in the file name, they are resolved at this time.

Managing File Names

This type of requirements means that the Process Record algorithm needs to provide a file name that has some business value to distinguish the individual files. For example, the service provider or the division or the person type value must be part of the file name. However, the recommendation is not to have the file name entirely determined by the algorithm. Rather the suggestion is that you still allow the user submitting the batch job to provide a desired file name as a batch parameter. That way, it can include any of the supported substitution variables for system information (such as batch run number, or date /time stamp) that ensure your files are unique for a given batch run. The algorithm can designate a substitution variable that it looks for to replace with the business value.

For example, using the files per division use case, imagine the user indicates the file name EXTRACT_FILE_<DIV>_{BN}_{RN}_{TN}.csv. The process record algorithm receives this parameter. It knows to look for "<DIV>" and replace it with the actual division. Imagine the division code is A100, it would return the file name EXTRACT_FILE_A100_{BN}_{RN}_{TN}.csv in its output, along with the data to write to that file for this unit of work.

For the person information use case above, where two files are written, the suggestion is to use the File Name parameter for one of the files, for example the person information file. An additional parameter should be used to define the second file, the contact information file. So the users submitting the batch job would define the following parameters:

Parameter Name	Description	Value	Comments
fileName	File Name	persons_<personType>_{BN}.txt	This is the file name parameter provided with the extract.
contactFileName	Contact File Name	contacts_{BN}.txt	This is a special parameter defined by the CM extract batch control.

The Process Record algorithm would look for the File Name parameter provided as input, find the text "<personType>" and replace it with the person type value and return that file name in the output. When building the list of schemas to return in the output, the one that includes the person contact information would include the file name found in the Contact File Name parameter provided as input.

Note: If an extract uses file integration types, the file integration record's extract process algorithm would have the above responsibility for indicating the file name and pass it back to the Process Record algorithm.

Multiple Threads

This functionality works as expected with multiple threads. Based on the use case and the implementation of the Select Records algorithm, each thread could be producing files for the same business value. Let's use the division use case. Imagine an implementation has three divisions: A100, B200, C300. Now imagine that the extract is for financial information for an account. The select records algorithm should be threading by account id so that the data is evenly

Copyright © 2024, Oracle and/or its affiliates.

distributed across the multiple threads. But within each thread, accounts could be in any of the 3 implementation's divisions. So each thread could produce 3 files for each of the 3 divisions.

As described in Multi-threaded Extract, the system supports an option to concatenate, allowing you to consolidate the files such that you have 3 complete files for the 3 divisions. The same features and limitations described in that topic apply for this use case.

Closing Files

An important consideration when using this functionality is that attention should be paid to how many files are open during the batch process. Each open file takes some memory to manage it. As such, the product has imposed a limit of only 10 open files (per thread) during the run of the batch job. When designing a batch to write to multiple files, the product strongly encourages you to design the process so that whenever possible, all the data for one file can be written and that file can be closed before going on to the next file.

Let's look at our examples.

- For the service provider example, the unit of work for the batch process is service provider. In this case, after each call to process records, the data can be written to the file and then the file can be closed.
- For the division example, the data selected should be threaded by the proper unit of work related to the purpose of the extract, such as Account or Asset. But the data should be ordered by division so that all the data for each division is processed before the next division's data. Each division's file can be closed once the next division's data starts getting processed.
- For the person extract example, the contact file is used for all the data so that file needs to remain open throughout the duration of the job. However, the person information gets segrated by person type. The Select Records could order the data by person type so that all the data for one type could be written and then all the data for the other type could be written. However, given that for any run no more than 3 files will ever be open, this use case is one where there is no need to actively manage closing files during the process.

How will the batch program know when to close a file? Since the batch program is agnostic and does not know what type of data is being processed nor when files should be closed, the Process Record algorithm should provide this information.

> **Note:** The batch process will issue an error if Process Records indicates that the file to write to is one that was already closed. Reopening closed files is not allowed. The design of the process needs to be reviewed if this error is received.

To manage this, the batch process provides the Process Record algorithm with the list of open files (with the system substitution variables unresolved) and a Boolean to mark whether the file should be closed.

> **Note:** If an extract uses file integration types, the file integration record's extract process algorithm receives this list of open files and would have the responsibility described here to manage closing files.

To explain further, we'll use our examples.

File Per Division Example

This use case is that there are 4 divisions in the implementation: NORTH, EAST, SOUTH, WEST and that the data (account data) should be grouped in separate files by division.

The recommendation is that the Select Records algorithm selects the data ordered by Division.

The batch process passes to the Process Record algorithm the list of open files (there should be only be zero or one). In this case, the only indication that the data for one division is finished is if the Process Record algorithm sees that its data is for one division and the open file listed is for a different division. Each call should prepare its file name. Then it should see if its file name is in the list of open files. If not, it should indicate that the other open file should be closed. (Note that it doesn't need to add itself to the 'open list'. The batch program is only looking for entries marked as closed. It will ignore entries that are not marked as closed).

Assume the file name in the batch parameter is myFile_<DIV>_{BN}.txt

Copyright © 2024, Oracle and/or its affiliates.

The data is sorted so that the accounts in the EAST division are processed first. Let's imagine this is a work unit that is also in the EAST division. It receives the file name and the list of open files.

```
...
<hard>
<batchParms>
  <BatchParm>
    <name>fileName</name>
    <value>myFile_<DIV>_{BN}.txt</value>
  </BatchParm>
  -...
</batchParms>
<openFiles>
  <OpenFile>
    <closeFile></closeFile>
    <fileName>myFile_EAST_{BN}.txt</fileName>
  </OpenFile>
</openFiles>
...
</hard>
```

The algorithm substitutes its division into the <DIV> portion of the file and populates that in the output. It then compares its file name with the current list of open files. It finds its file so it doesn't need to do anything else.

```
<hard>
...
<fileName>myFile_EAST_{BN}.txt</fileName>
...
<openFiles>
  <OpenFile>
    <closeFile></closeFile>
    <fileName>myFile_EAST_{BN}.txt</fileName>
  </OpenFile>
</openFiles>
...
</hard>
```

If we then imagine a call to the process records algorithm where the division is NORTH and it's the first entry for that division, here is what it receives:

```
<hard>
<batchParms>
  <BatchParm>
    <name>fileName</name>
    <value>myFile_<DIV>_{BN}.txt</value>
  </BatchParm>
  -...
</batchParms>
<openFiles>
  <OpenFile>
    <closeFile>true</closeFile>
    <fileName>myFile_EAST_{BN}.txt</fileName>
  </OpenFile>
</openFiles>
...
</hard>
```

The algorithm substitutes its division into the <DIV> portion of the file and populates that in the output. It then compares its file name with the current list of open files and doesn't finds its file name. So it indicates that the existing open file should be closed.

```
<hard>
...
<fileName>myFile_NORTH_{BN}.txt</fileName>
```

Copyright © 2024, Oracle and/or its affiliates.

```
...
<openFiles>
  <OpenFile>
    <closeFile>true</closeFile>
    <fileName>myFile_EAST_{BN}.txt</fileName>
  </OpenFile>
</openFiles>
...
</hard>
```

File Per Service Provider Example

In this use case, the unit of work is service provider and each service provider should have its own file. In this case, as the Process Record algorithm is preparing the data and the file name unique to its service provider, it can also indicate that the file can be closed. There is no need to wait for the next call to Process Record to indicate that the previous file can be closed. Let's assume the file name is MYFILE_<serviceProvder>_{BN}_{TN}.txt. Each call to Process Records will pass the batch parm file name and no open files.

```
<hard>
<batchParms>
  <BatchParm>
    <name>fileName</name>
    <value>MYFILE_<serviceProvider>_{BC}_{TN}.txt</value>
  </BatchParm>
  -...
</batchParms>
<openFiles>
  <OpenFile>
    <fileName></fileName>
    <closeFile></closeFile>
  </OpenFile>
</openFiles>
...
</hard>
```

The algorithm is processing data for service provider A100. It returns the following information related to the file name, namely the single File Name in the output related to this service provider and adds its file to the open file list and indicates that it should be closed. (Note that file-per-work-unit is the only known use case where the algorithm will know that it is the last entry for a file and to close itself.)

```
<hard>
...
<fileName>MYFILE_A100_{BC}_{TN}.txt</fileName>
...
<openFiles>
  <OpenFile>
    <closeFile>true</closeFile>
    <fileName>MYFILE_A100_{BC}_{TN}.txt</fileName>
  </OpenFile>
</openFiles>
...
</hard>
```

Multiple Open Files Example

This example assumes that the file names are defined as per the Managing File Names section above. We'll assume that the algorithm designer chose to not actively close the person information file since there are never more than three files open. Similarly, we'll assume that they chose not to order the data by person type. Here is what is passed in in the middle of the run:

```
<hard>
<batchParms>
  <BatchParm>
    <name>fileName</name>
    <value>persons_<personType>_{BN}.txt</value>
```

Copyright © 2024, Oracle and/or its affiliates.

```
  </BatchParm>
  <BatchParm>
    <name>contactFileName</name>
    <value>contacts_{BN}.txt</value>
   </BatchParm>
  -...
</batchParms>
<openFiles>
  <OpenFile>
    <fileName>persons_P_{BN}.txt</fileName>
    <closeFile></closeFile>
  </OpenFile>
   <OpenFile>
    <fileName>persons_B_{BN}.txt</fileName>
    <closeFile></closeFile>
  </OpenFile>
  <OpenFile>
    <fileName>contacts_{BN}.txt</fileName>
    <closeFile></closeFile>
  </OpenFile>
</openFiles>
...
</hard>
```

An instance of one call to Process Records algorithm that has details for an individual ("P" person type).

```
<fileName>persons_P_{BN}.txt</fileName> -
<fileOutput>
  <listValue>
    <SchemaInstance>
      <recordXMLNode>person</recordXMLNode>
      <schemaName>CM-PersonDetail</schemaName>
      <schemaType>F1SS</schemaType>
      <fileName></fileName>
      <data>LotsOfData</data>
    </SchemaInstance>
  </listValue>
  <listValue>
    <SchemaInstance>
      <recordXMLNode>person</recordXMLNode>
      <schemaName>CM-AddressDetail</schemaName>
      <schemaType>F1SS</schemaType>
      <fileName></fileName>
      <data>LotsOfData</data>
    </SchemaInstance>
  </listValue>
  <listValue>
    <SchemaInstance>
      <recordXMLNode></recordXMLNode>
      <schemaName>CM-ContactDetail</schemaName>
      <schemaType>F1SS</schemaType>
      <fileName>contacts_{BN}.txt</fileName>
      <data>LotsOfData</data>
    </SchemaInstance>
  </listValue>
</fileOutput>
```

This XML example illustrates the following points:

- There is an override file name in the fileName element.
- The first two schemas don't have any value in the fileOutput\listValue\SchemaInstance\fileName so they should be written to the file in fileName. Note that these two schemas also include a Record XML Node. That will ensure that if the output is XML format, the data from both of those schemas will be wrapped by the 'person' XML Node. This is not functionality related to this epic. It's just trying to illustrate multiple features.
- The third schema has an entry in fileOutput\listValue\SchemaInstance\fileName . So that data is written to a separate file.

Copyright © 2024, Oracle and/or its affiliates.

Extract Using File Integration

The product provides separate configuration that may be used to define how to extract the data and how to format it. This configuration provides more metadata related to defining how to get the data to extract and how to format the data. You may have many use cases where using this configuration rather than having all the logic directly in the Process Records algorithm is beneficial.

The configuration can support simple extracts, containing a single record type and possibly a header or footer record. It also supports more complex integrations, involving multiple record types and parent/child relationships between the records.

File Integration Type Configuration

The framework supplies two objects that work together to define the processing needed to create a specific type of extract:

- The file integration type which describes and classifies the extract configuration.
- The file integration record where the sequence, format and logic to extract the data required for a given record type are captured.

File Integration Type

The file integration type defines the integration classification. Currently, only **Extract** is supported. In addition, the file integration type can be assigned to a category. The category may be used to identify extracts with a common purpose or target such as collection agency referrals. Your edge product may deliver a number of categories or you may define your own.

The system provides a business object for extract file integration types (**F1-ExtractFileIntegrationType**) that is expected to be used for most use cases

File Integration Record

The file integration record is the object that is responsible for defining a record's format and capturing the record details. A separate integration record must be defined for each record type included in the extract.

The record sequence defines the order that records are processed by the system. The parent sequence defines the record type of which this record is a child. Child records may also be referenced as a parent of lower level child records. The base Process Records algorithm provided by the framework processes all child records of a record type before processing the next record type in sequence.

For extracts that are XML format, the system supports creating a grouping node surrounding all the output that belongs to the same entity. File integration records define a record XML node that will be used as the grouping node. Each record type in the list that belongs within the same group would indicate the same value in the record XML node. Refer to the Grouping by Record XML Node section for more details.

Note: The system does not support nested record XML nodes. All child records of a record type must define the same record XML node as the parent.

Each file integration record references a data area that defines the extract record's format, allowing the data to be extended. The **Extract Records** algorithms are responsible for populating the data area. The file integration record supports multiple extract processing algorithms so that the custom information can also be populated.

The extract record plug-in also supports the functionality described in the Flexible File Name / Writing Multiple Files topic.

The system provides a business object for extract file integration records (**F1-ExtractFileIntegrationRec**) that is expected to be used for most extract record use cases.

Copyright © 2024, Oracle and/or its affiliates.

Configuring an Extract Process Using File Integration Type

The following points summarize the additional steps needed to implement a new extract background process that uses a file integration type to define the specific functionality:

- Choose or create data area objects to define the extract record formats.
- Define the logic required to populate each extract record type. Your edge product may deliver suitable plug-ins or you may need to create a plug-in script where the algorithm entity is **File Integration Record - Extract Record**. Note that the Extract Record plug-in receives most of the information defined in the Batch Control - Process Records plug-in.
- Define the algorithm type and algorithm for each newly created script.
- Choose or create a category for the file integration type using lookup **FILE_INT_OBJ_FLG**.
- Create the file integration type and configure the file integration records using the defined data areas and algorithms.
- Create a batch control by duplicating the appropriate base template for an extract background process. Include the File Integration Type parameter (fileIntegrationType) on the batch control and reference the appropriate file integration type.
- Plug in a Process Records algorithm that reads the file integration type configuration and processes the record types. The system provides a base file integration record extraction algorithm **(F1-FILEX-PR)** that is expected to be used for most file integration plug-in extract processes.

Extending a File Integration Extract Process

Your edge product may deliver file integration types and batch controls designed for standard extract processes. However, extending the delivered extracts to add custom data is common practice for most implementations. You can customize the output from a delivered extract by:

- Identifying the record types that need additional details
- Creating a data area that defines the additional details and using that to extend the data area configured on the record type
- Creating a File Integration Record - Extract Record plug-in script, algorithm type and algorithm that populates the extra details
- Adding the algorithm to the integration record's extract algorithms collection

In some cases, you may wish to extend the extract by adding additional record types. To do this, add your custom file integration records to the file integration type, following the steps described in Configuring an Extract Process Using File Integration Type.

Uploading Records

The base product provides a background process to upload data from a file. The batch control Plug-in Driven File Upload Template (**F1-PDUPL**) may be used as a template. The process includes parameters to configure the file path and file name for file to upload along with other parameters to control how to handle missing files and how to rename the file once processed. Refer to the description of the batch control and its parameters for more information.

This background process requires an algorithm plugged into the plug-in spot **File Upload**. This plug-in is called once for a given file. The batch process opens the file and passes to the algorithm the file reader element. The algorithm associated with the batch control is responsible for using provided APIs to read the content of the file and store the data in appropriate table(s) (for example, an appropriate staging table). The base provided process supports uploading multiple files and may be run mutli-threaded to process multiple files. Each file is processed by one call to the File Upload algorithm and supports a single commit for the records uploaded in a given file.

Note: Plug-in scripts written to implement this type of algorithm must use the Groovy script engine version as the APIs are not accessible using the XPath scripting language. Refer to Additional Coding Options For Server Scripts for more information.

Note that this step in the upload of data is only one part of a typical upload end-to-end process. This step is sometimes referred to in the product as "Process X". The goal of this step is to get records from a file into database records, with minimal validation and processing. The data should be stored in records that are then processed by a second step (often

Copyright © 2024, Oracle and/or its affiliates.

referred to in the product as the "upload" step, for example "Payment Upload"). The second step, independent of the plug-in driven batch process described here, is responsible for validating the data and should be able to be threaded by individual records and have proper error handling for individual records. Note that depending on the type of data being uploaded, the product may already supply appropriate tables that the plug-in driven upload process may populate. These could be staging tables, such as payment upload staging. Or they may be records with business objects that have a lifecycle designed to handle uploaded data, for example Business Flag. In such cases, the product will typically supply out-of-the-box background processes to validate and further process the data and finalize the upload. If the data to upload does not already have a base provided staging table, be sure to work with your implementation team to identify an appropriate table to use for the plug-in driven batch upload. In addition, confirm the design for the second step that is responsible for the detailed validation and finalization of the data.

The product supplies sample algorithms to illustrate calling the supplied APIs for processing different types of source data: comma delimited, fixed position and XML. In every case, the sample data supported by the upload uses 'degree day' information to illustrate the process. The system provides sample target records (based on the Fact maintenance object) in order to illustrate the step to store records based on the input data. Note that only sample plug-in scripts have been provided. No algorithm, algorithm type or batch control is configured to use the sample plug-in scripts. To view the scripts, navigate to the Script page, search for a Script Type of **Plug-in Script** and Algorithm Entity of **Batch Control - File Upload** and look for the 'Sample' scripts.

Note that the sample plug-in scripts provided by the product are supplied to illustrate use of the provided APIs. They should not be considered a template for how to implement a plug-in script for a real use case. The following highlight some points to consider when designing a file upload algorithm:

- **Error handling / resolution.** The sample plug-in scripts do some basic error handling related to the data to illustrate error handling. However, any errors found in this step require processing of the whole file to stop. As such, this plug-in should only report errors that are not possible to fix, but where the whole file should be rejected. If there are errors that can be adjusted in the data, then the recommendation is to not check for those errors at this step. Rather, this plug-in should simply populate the appropriate staging tables and let the next step check for validity. As described above, the next step should include the ability to mark individual records in error, allowing for users to fix the data and retry.
- **Target tables.** The sample plug-in scripts use Fact as the target for the resulting insert statements. As mentioned above, the decision of where to store the uploaded data must be carefully considered. There may already be existing tables that are specific to a given use case. If the data being uploaded does not have existing tables to use, review the product to verify what existing tables may be useful, such as Inbound Sync Request or Service Task. Be sure that the tables chosen support error handling, either out-of-the box or via designing an appropriate business object with a lifecycle that supports an error status and the ability to resolve the error. Also note that the Sample Flat File Upload plug-in illustrates a header record / detail record scenario. In this case, the header record is linked to the child record via a CLOB element. This is not the recommended technique. In a real use case, the header record should be linked to the child record via a separate database column to allow for searching.

Configuring a New Process

The following points summarize the steps needed to implement a new file upload background process:

- Verify the details of the data in the upload file and map the data to fields in one or more appropriate tables in the system.
- Design the logic required for reading the record details and identifying each record to properly create the insert statements for storing the data. The sample plug-in scripts provided by the product illustrate using the various APIs available for use. Create a plug-in script where the algorithm entity is **Batch Control - File Upload**. Create an appropriate algorithm type and algorithm for this plug-in script.
- Create a batch control by duplicating the base template. Plug in the algorithm created above and configure the parameters as appropriate. Note that you may configure custom ad hoc parameters on the batch control if required. Both base and custom batch parameter values are available to the File Upload algorithm.

How to Re-extract Information

If you need to recreate the records associated with an historical execution of an extract process, you can - simply supply the desired batch number when you request the batch process.

Copyright © 2024, Oracle and/or its affiliates.

How to Submit Batch Jobs

Most batch jobs are submitted via a batch scheduler. Refer to Batch Scheduler Integration for information about the integration with the Oracle Scheduler.

Batch jobs may be configured as Timed, which means they will automatically be run based on the timer frequency.

You can manually submit your adhoc background processes or submit a special run for one of your scheduled background processes using the online batch job submission page.

The system provides a REST web service (using the inbound web service **F1-SubmitJob**) to create a batch job submission request from an external system. Note that no parameter defaulting occurs when submitting a record using a web service. All the desired parameter values must be provided.

How to Track Batch Jobs

You can track batch jobs using the batch process pages, which show the execution status of batch processes. For a specified batch control id and run id, the tree shows each thread, the run-instances of each thread, and any messages (informational, warnings, and errors) that might have occurred during the run.

Fastpath: For more information, refer to Tracking Batch Processes.

How to Restart Failed Jobs and Processes

Every process in the system can be easily restarted if it fails (after fixing the cause of the failure). All you have to do is resubmit the failed job; the system handles the restart.

Assessing Level of Service

For some background processes, an implementation may wish to supply one or more algorithms that check some conditions to assess whether or not the process is performing as expected. An algorithm could be used to check the performance of the job to see if it is running as efficiently as expected. Or it could analyze the data processed by the background process to assess whether there may be some problem with the quality of the data.

The system provides a Level of Service plug-in spot on batch control to configure the appropriate algorithms for a given background process, if desired. Each algorithm is expected to return a value to indicate the 'level of service' determined along with a message indicating the reason for the value. The following Level of Service values are supported:

- **Normal**. Indicates that the algorithm did not detect any issues.
- **Warning**. Indicates that the algorithm found some issues that may or may not indicate a problem.
- **Error**. Indicates that the algorithm found some issues that should be investigated.
- **Disabled**. Indicates that the algorithm could not properly execute the level of service logic.

When viewing a batch control record, if there are any level of service algorithms configured, the logic is executed and the results are displayed. The level of service is also part of the Health Check service.

Batch Resource Groups

Database resource management tools may be used to prioratize resource allocation between various batch processes. For example, export type of processes may have a lower resource allocation priority relative to billing related processes. These tools allocate resources to groups of consumeers, not to an individual batch process. They use rules that are based on certain session variables, to associate a batch process with a resource group.

Copyright © 2024, Oracle and/or its affiliates.

When applicable, you may use the **Batch Resource Configuration (F1-BatchResourceConfig)** extendable lookup to associate certain batch processes with one of three priority resource groups: 10 (High), 20 (Medium), 30 (Low).

In a Cloud installation, if a batch control is associated with a resource group the latter is included by default as part of the MODULE database session variable as follows: '<batch control>,R=<resource group>'. This allows DBAs to potentially take advantage of this information for resource prioritization purposes. On premise, this feature needs to be explicitly opted in using the **Expose Batch Resource Group** option in the **General System Configuration** feature configuration.

With this configuration enabled, DBAs can set up rules that parse the MODULE variable to identify the resource group for a batch process and apply prioritization rules accordingly. Note that most batch controls may not be associated with a resource groups so default allocation rules should be considered.

Note. This configuration does not include nor enforce the database resource management configuration. It only allows for such tool to be leveraged by DBAs as needed.

Defining Batch Controls

The system is delivered with all necessary batch controls. Implementations may define default values for parameters. In addition, implementations may define their own background processes.

To view background processes, open **Admin** > **System** > **Batch Control.**

Use the **Batch Control Query** portal to search for an existing batch control. Once a batch control is selected, you are brought to the maintenance portal to view and maintain the selected record.

Refer to Background Processing Concepts for more information.

CAUTION: Important! If you introduce a new batch process, carefully consider its naming convention. Refer to System Data Naming Convention for more information.

Description of Page

Enter an easily recognizable **Batch Process** and **Description** for each batch process.

Owner indicates if this batch control is owned by the base package or by your implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when you add a batch control. This information is display-only.

Use the **Detailed Description** to describe the functionality of the batch process in detail.

Enter the **Application Service** that is used to provide security for submission requests for the batch control. The application service must have an Access Mode of **Execute**. Refer to Granting Access To Batch Submission for more information.

Use **Batch Control Type** to define the batch process as either **Timed** or **Not Timed**. A Timed batch process will be automatically initialized on a regular basis. A Not Timed process needs to be run manually or through a scheduler.

Use **Batch Control Category** to categorize the process for documentation purposes. The base values provided are as follows:

- **Ad Hoc**. Processes of this type are run on an ad hoc basis, only when needed. For example, if there is a process to do a mass cancel / correction of data, it would only be run when a situation occurs requiring this.
- **Extract**. Extract processes extract information that is interfaced out of the system. Processes of this type typically extract records marked with a given run number. If the requester of the process does not supply a specific run number, the system assumes that the latest run number should be extracted. If you need to re-extract an historical batch, you simply supply the respective run number when you request the batch process.
- **ILM**. Information Lifecycle Management jobs are crawler background processes that are associated with the ILM based storage solution.
- **Monitor**. Processes of this type are processes related to business objects with a lifecycle state that defines monitor algorithms. The monitor process selects records in a given state and executes its algorithms, which may cause the record to transition to another state or may trigger some other logic to occur. Using configuration, the monitor process

Copyright © 2024, Oracle and/or its affiliates.

may target only specific records. Refer to Monitoring Batch Processes for more information. Note that these types of background processes can be considered a subset of **Process What's Ready**

- **Process What's Ready**. Processes of this type create and update records that are "ready" for processing. The definition of "ready" differs for every process. For example, a payment upload process creates payments for every record that is **pending**. An overdue event monitor activates pending overdue events that have reached their trigger date.
- **Purge**. Processes of this type are used to purge historical records from certain objects that generate a large number of entries and may become unwieldy over time.
- **To Do Entry**. Processes of this type are used to detect a given situation and create or complete a To Do Entry. Refer to To Do Entries Created by Background Processes for more information.
- **Upgrade**. Processes of this type are typically run only one time and are used to populate data in a table based on information that can be determined elsewhere in the system. These are processes that could have been done through an upgrade script, but perhaps due to data volume or timing, the upgrade logic warranted a separate stand-alone process.
- **Upload**. Processes of this type are used to upload data from a file into the system.
- The following categories are related to the data conversion / migration processes:
  - **Conversion**. Processes of this type are dedicated to converting or migrating data from external applications into the product.
  - **Object Validation**. Processes of this type are dedicated to validate data within objects for conversion or migration purposes.
  - **Referential Integrity**. Processes of this type are dedicated to validate referential integrity within objects for conversion or migration purposes.

Note: Additional categories may be introduced by your specific product.

If the batch process is Timed, then the following fields are available:

- **Timer Interval** is the number of seconds between batch process submissions. The system will start the next run this many seconds from the start time of the previous run.
- **User ID** is the ID under which the batch process will run.
- **Email Address** is the Email address to be used for notification if the batch process fails.
- **Timer Active** allows you to temporarily switch off the timer while retaining the other settings for the timed job.
- **Batch Language** is the language associated with the batch process.

Use **Program Type** to define if the batch process program is written in **Java** or **Java (Converted)**, meaning that the program has been converted into Java.

Note: Java (Converted) program types are not applicable to all products.

Use **Program Name** to define the Java class / program associated with your batch process.

Note: View the source. If the program is shipped with the base package, you can use the adjacent button to display the source code of this program in the Javadocs viewer.

Level of Service shows the output of the level of service algorithms for the batch control. If one algorithm is plugged into the Batch Control, the level of service lookup value along with a message indicating the reason for the output value is shown. If multiple algorithms are plugged in, the text **See Results for Details** is displayed. There is an icon provided to expand the details returned by each algorithm. If no level of service algorithm is found, then the value **Disabled** is shown with a message indicating that no algorithm is provided for this batch control.

The **Last Update Timestamp, Last Update Instance** and **Next Batch Nbr** are used for audit purposes.

Turn on **Accumulate All Instances** to control how this batch control is displayed in the Batch Run Tree. If checked, the run statistics (i.e., "Records Processed" and "Records in Error") for a thread are to be accumulated from all the instances of the thread. This would include the original thread instance, as well as any restarted instances. If this is not turned on, only the ending (last) thread instance's statistics are used as the thread's statistics. This may be preferred for certain

Copyright © 2024, Oracle and/or its affiliates.

types of batch processes where the accumulation would create inaccurate thread statistics, such as those that process flat files and, therefore, always start at the beginning, even in the case of a restart.

The following fields are default values that are used when a batch job is submitted for the batch control:

- Use **Thread Count** to control whether a background process is run single threaded or in multiple parallel threads. This value defines the total number of threads that have been scheduled.
- Select **Trace Program Start** if you want a message to be written whenever a program is started.
- Select **Trace SQL** if you want a message to be written whenever an SQL statement is executed.
- Use **Override Nbr Records to Commit** to define the default number of records to commit. This is used as the default value for timed jobs as well as online submission of jobs that are not timed.
- Select **Trace Program Exit** if you want a message to be written whenever a program is exited.
- Select **Trace Output** if you want a message to be displayed for special information logged by the background process.

For more information about these fields, see Batch Job Submission - Main

The parameter collection is used to define additional parameters required for a particular background process. The following fields should be defined for each parameter:

Sequence. Defines the relative position of the parameter.

Parameter Name. The name of the parameter as defined by the background process program.

Description. A description of the parameter.

Detailed Description. A more detailed description of the parameter.

Required. Indicates whether or not this is a required parameter.

Parameter Value. The default value, if applicable. Any value entered here is displayed to a user when submitting a job through online Batch Submission. Users may override the value before submitting the job. This includes resetting the value for an optional parameter. Regardless of how a batch job is submitted, if a value is not supplied for a required parameter, the default value define here, if found, is used. Optional parameters with no value supplied at submission time will not be defaulted to a value defined here. Note that an implementation may define a default value for base provided batch controls.

Security. Indicates whether the system should **Encrypt** the parameter value or not. A value of **Encrypt** means that the parameter value is stored in the database and written to the log files using encryption. In addition, the parameter is written to the log files with asterisks. The setting applies to values entered here as well as in the online Batch Submission. If there is no need to secure the parameter value, use the default setting of **None**.

Owner Indicates if this batch process is owned by the base package or by your implementation (Customer Modification). The system sets the owner to **Customer Modification** when you add a batch process. This information is display-only.

- Batch Control - Algorithms on page 417

Batch Control - Algorithms

Use this page to maintain a batch control's algorithms. Open this page using **Admin** > **System** > **Batch Control** and then navigate to the **Algorithms** tab.

Description of Page

The **Algorithms** grid contains algorithms that control important functions for instances of this batch control. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence Number** and **Algorithm** for each system event. You can set the **Sequence Number** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the **Sequence** in which they should execute.

Copyright © 2024, Oracle and/or its affiliates.

- **Owner** indicates if this is owned by the base package or by your implementation (**Customer Modification**).

The following table describes the **System Events**.

System Event	Optional / Required	Description
Error Post-Processing	Optional	Algorithms of this type are called if the batch process fails to complete due to an error. Multiple algorithms are allowed and are executed in sequence order. Refer to Error Post-Processing Logic for more information.
File Upload	Optional	Algorithms of this type are only applicable to plug-in driven background processes and are used to upload all records for a file. This is a single algorithm plug-in spot. If multiple algorithms are plugged into the batch control, the one with the highest sequence is used.
Level of Service	Optional	Algorithms of this type are called to determine the Level of Service provided by a batch control. Multiple algorithms are allowed. Refer to Assessing Level of Service for more information.
Post-Processing	Optional	Algorithms of this type are called after all threads are complete. Multiple algorithms are allowed and are executed in sequence order. Refer to Post-Processing Logic for more information.
Process Record	Optional	Algorithms of this type are only applicable to plug-in driven background processes and are used to process a specific record. This is a single algorithm plug-in spot. If multiple algorithms are plugged into the batch control, the one with the highest sequence is used.
Select Records	Optional	Algorithms of this type are only applicable to plug-in driven background processes and are used to define the SQL to use to select the records to process. This is a single algorithm plug-in spot. If multiple algorithms are plugged into the batch control, the one with the highest sequence is used.

Note: You can add new system events. Your implementation may want to add additional batch control oriented system events. To do that, add your new values to the customizable lookup field **F1_BATCH_CTRL_SEVT_FLG**.

Defining File Integration Types

This section describes the functionality supported for viewing and maintaining file integration types.

Refer to Extract Record Processing for an overview of the file integration type functionality provided with plug-in driven background processes.

To view or add a file integration type, open **Admin** > **Integration** > **File Integration Type.**

Copyright © 2024, Oracle and/or its affiliates.

Once a file integration type record has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The **File Integration Records** zone lists the record definitions for this integration, in hierarchical order. Use the zone actions to add, maintain or view a file integration record. Press the **Record Layout** icon to view a pop-up display of the record's schema.

On-Line Batch Submission

The on-line batch submission page enables you to request a specific background process to be run. When submitting a background process on-line, you may override standard system parameters and you may be required to supply additional parameters for your specific background process. After submitting your background process, you may use this page to review the status of the submission.

The following topics further describe logic available for on-line submission of background processes.

Batch Submission Creates a Batch Run

When you request a batch job to be submitted from on-line, the execution of the desired background process will result in the creation of a batch run. Just as with background processes executed through your scheduler, you may use the Batch Run Tree page to view the status of the run, the status of each thread, the run-instances of each thread, and any messages that might have occurred during the run.

Note: Your on-line submission record is assigned a status value so that you may know whether your job has been submitted and whether or not it has ended, however, it will not contain any information about the results of the background process itself. You must navigate to the Batch Run Tree page to view this detail.

Jobs Submitted in the Background

When you save a record on the batch job submission page, the batch job does not get submitted automatically. Rather, it saves a record in the batch job table. A special background process will periodically check this table for pending records and will execute the batch job. This background process will update the status of the batch job submission record so that a user can determine when their job is complete.

Note: At installation time, your system administrator will set up this special background process to periodically check for pending records in the batch job submission table. Your administrator will define how often the system will look for pending records in this table.

It should be noted that this special background process only submits one pending batch job submission record at a time. It submits a job and waits for it to end before submitting the next pending job.

Note: If you request a batch job to be run multi-threaded, the special background process will submit the job as requested. It will wait for all threads to complete before marking the batch job submission record as **ended**. Refer to Running Multi-threaded Processes for more information.

Email Notification

If you wish the system to inform you when the background process completes, you may supply your email address. The email you receive will contain details related to the batch job's output; similar to the job results you would see from your batch scheduler.

The email includes information from the product installation to help users that may be working in and submitting background processes in more than one environment.

Copyright © 2024, Oracle and/or its affiliates.

- It includes the product name and product version for the product in the installed products list that is marked as the one to display.
- If a domain name message is configured in the installation messages, this information is also included.

Note: This assumes that during the installation process, your system administrator configured the system to enable email notification. Your administrator may also override the amount of detail included in the email notification.

Note: A batch control parameter may be used to indicate if the email should only be sent in the case of an error or success or both. Refer to Parameters Supplied to Background Processes for more information.

Running Multi-Threaded Processes

Many of the system background processes may be run multi-threaded. When submitting a background process on-line, you may also run a multi-threaded process. The fields Thread Count and Thread Number on the batch submission page control the multi-threaded process requests:

- To run a multi-threaded process, indicate the number of threads in **Thread Count** and enter **0** in the **Thread Number**. If a previous run ended with one or more threads in error, a new submission may be created to restart all the threads that were in error. This submission would result in new Batch Thread Instance records added for each thread that previously ended in error. Note that if the previous run was marked to "Do Not Restart", then a new submission will result in a new Batch Run.
- In the situation where a previous run resulted in one or more threads in error, you may use the Thread Number to indicate a specific thread to restart. This is only valid if there is a previous run found that is not marked "Do Not Restart". This will result in a new Batch Thread Instance created for the specific thread in the previous run.
- To run a process as a single thread, enter Thread Count = **1** and Thread Number = **1** (or **0**). This will execute the background process single-threaded.

Note: When running a multi-threaded process, the special background process will wait until all threads are complete before marking the batch job submission record as **Ended**.

Batch Jobs May End in Error

It is possible for your background process to end with an error. When this occurs, your batch job submission record will still be marked as **Ended**. You will need to navigate to the Batch Run Tree page to determine the status of the batch run.

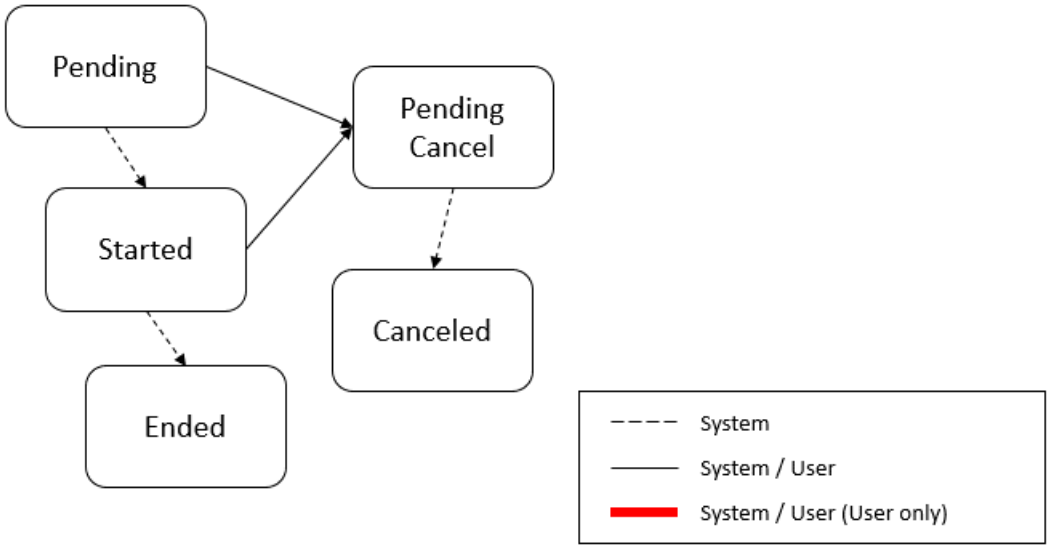
Submitting Jobs in the Future

If you wish to request a batch job to be submitted in the future, you may do so when creating your batch job submission record by entering a future submission date. The special background process, which looks for pending records in the batch job submission table, will only submit batch jobs that do not have a future submission date.

Lifecycle of a Batch Job Submission

The following diagram illustrates the lifecycle of a batch job submission record.

Copyright © 2024, Oracle and/or its affiliates.



Pending – Records are created in **Pending** status. Records in this state are put in a queue to be submitted.

Started – Once a pending record has been submitted for processing, its status will be changed to **Started**.

Pending Cancel – Users may request the cancellation of a pending record to prevent the batch job from being submitted. In addition, users may request to cancel a started batch job that was not initiated by DBMS scheduler. The system will look for records in this status and look for batch runs that may need to be canceled for this batch job. Once it is confirmed that any batch runs are stopped, the status changes to **Canceled**.

Ended – When the batch job has finished processing, its status will be changed to **Ended**. Note that records in **Ended** status may have ended in error. Refer to Batch Jobs May End in Error for more information.

Granting Access To Batch Submission

Every batch control must reference an application service. When you link a batch control to an application service, you are granting all users in the group the ability to submit the associated batch job for execution.

Fastpath: Refer to The Big Picture Of Application Security for information about granting users access rights to an application service.

When the batch control is added to the batch job table, the system checks if both the user submitting the job and the user ID recorded on the batch submission record have access rights. Application security does not apply when viewing a batch control or an associated batch run.

Note: Base batch controls will be associated with base owned application services by default. Implementations will need to ensure the appropriate user groups are linked to these application services.

Batch Job Submission Portal

Refer to On-Line Batch Submission for detailed explanations about many feaures of online submisssion.

To view a submitted batch job, use **Menu** > **Tools** > **Batch Job Submission**. You are brought to a query portal with options for searching for a Batch Job Submisssion entry.

Once a batch job submission entry has been selected, you are brought to the maintenance portal to view the selected record.

Copyright © 2024, Oracle and/or its affiliates.

The inline help provides information about several fields. The following points highlight information about some of the fields that may be visible.

The **Batch Job Status** indicates the current status of the batch job. Refer to Lifecycle of a Batch Job Submission for more information.

Batch Run information will be displayed if the batch job submisssion has an associated run number. Click the link to navigate to the Batch Run tree.

Batch Business Date is only used for background processes that use a date in their processing. For example, a billing process can use the business date to determine which bill cycles should be downloaded. If this parameter is left blank, the system date is used at the time the background process is executed.

Note: Saving a record on this page does not submit the batch job immediately. A special background process will run periodically to find pending records and submit them. Depending on how often the special process checks for pending records and depending on how many other pending records are in the 'queue', there may be a slight lag in submission time. If the desired execution date/time is close to midnight, it is possible that your batch job will run on the day after you submit it. If you have left the business date blank in this case, keep in mind that your business date would be set to the day after you submit the job.

Batch User is the user ID associated with the run of the background process. It may differ from the user who submitted the batch, as described below. Refer to Parameters Supplied to Background Processes for more information about the significance of the user id.

Note: This field defaults to the ID of the current user when adding a new batch job submisssion entry.

Language is used to access language-specific control table values. For example, error messages are presented in this language code.

If you wish the system to notify you when the batch job is complete, enter your **Email Address**. Refer to Email Notification for more information.

Note: This field defaults to the email address for the current user, if populated on the user record.

The **Desired Execution Date/Time** defaults to the current date and time. Override this information if you wish the background process to be executed at some future date and time. Refer to Submitting Jobs in the Future for more information.

Thread Count is used to control whether a background processes is run single threaded or in multiple parallel threads. It contains the total number of threads that have been scheduled. For example, if the process has been set up to run in 20 parallel threads, each of the 20 instances receives a thread count of 20. Refer to Running Multi-threaded Processes for more information about populating this field.

Note: Not all processes may be run multi-threaded. Refer to the description of a batch control to find out if multi-threading is supported.

Thread Number should typically be set to 0. Refer to Running Multi-threaded Processes for situations where populating a value other than 0 are warranted.

The **Submission User** indicates who initiated the batch job. This value is not populated for timed batch jobs or batch jobs submitted via a scheduler. It is only displayed if the field is populated and the submission user is different to the batch user.

The **Submission Method** indicates how the batch job was initiated. It is visible in the **Record Information** section. Valid methods include:

- **Online** - populated when a user manually creates or manually duplicates a batch job submission.
- **Generated** - populated by algorithms that submit a batch job and by 'initiator' batch jobs that submit other batch jobs.
- **Scheduled** - populated by the DBMS scheduler. Note that for other schedulers, it is not advised to use this value as there may be logic and validation related to records created with this value that may not apply to other schedulers.
- **Timed** - populated by the batch daemon that submits timed batch jobs.

Copyright © 2024, Oracle and/or its affiliates.

- **Other** - populated when no other value is provided.

Additional values may be added by your implementation. Refer to the lookup **SUBMIT_METH_FLG** for a list of the submission methods supported by your system.

The following fields are shown in the **Additional Run Details** section:

Override Nbr Records To Commit and **Override Max Timeout Minutes**. These parameters are optional and override each background process's Standard Commit Records and Standard Cursor Re-Initiation Minutes. (Each background process's Standard Commit Records / Standard Cursor Re-Initiation Minutes should be documented in the detailed description of the batch control record). Note that Max Timeout Minutes corresponds to the Cursor Re-initiation Minutes.

Fastpath: Refer to Parameters Supplied to Background Processes for more information.

The following trace parameters may also be supplied to a background process and are only used during QA and benchmarking.

- **Trace Program Start** Turn on this switch if you wish a message to be written whenever a program is started.
- **Trace Program Exit** Turn on this switch if you wish a message to be written whenever a program is exited.
- **Trace SQL** Turn on this switch if you wish a message to be written whenever an SQL statement is executed.
- **Trace Output** Turn on this switch if you wish a message to be displayed for special information logged by the background process.

Note: The information displayed when the trace output switch is turned on depends on each background process. It is possible that a background process displays no special information for this switch.

Note: The location of the output of this trace information is defined by your system administrator at installation time.

If additional parameters have been defined for this background process on the Batch Control page, the **Parameter Name, Description, Parameter Value** and an indicator of whether or not the parameter is **Required** are displayed. A **Detailed Description** of the parameter can be displayed by clicking the arrow next to the description.

When adding a new batch job submisssion entry, if a default parameter value is configured on the batch control configuration, that value is shown and may be overridden. Note that if the parameter value is configured to be Encrypted on the batch control configuration, the value here will be shown encrypted.

Actions

This page has three special actions.

- Using the **Duplicate** button opens a window with the selected batch job submisssion details displayed. These fields, including the batch parameter values, may be overidden. Pressing **Save** will create a new Batch Job Submission entry.
- Use the **Rerun** button to rerun the selected batch job. A window opens displaying the details of the batch job submisssion to be rerun. These fields, including the batch parameter values, may be overidden. Pressing **Save** will create a rerun Batch Job Submission entry.

  Note: Rerun is only intended to be used for background processes that download information that belongs to given run number. It is typically only required if you need to download an historical run (rather than the latest run).
- **Cancel** is only applicable for batch job submisssions that are in **Pending** or **Started** status and do not have a submission method of **Scheduled** . The action is not visible for other statuses.

Tracking Batch Processes

The topics in this section cover the tools provided for reviewing the status and performance of batch processes. Refer to Defining Batch Controls for more information on how batch control codes are defined.

Copyright © 2024, Oracle and/or its affiliates.

Batch Run Tree

The batch run tree pages show the execution status of a selected batch process. For a specified batch control id and run id, the tree shows each thread, the run-instances of each thread, and any messages (informational, warnings, and errors) that might have occurred during the run.

Batch Run Tree - Main

This page allows you to view the status of a specific execution of a batch job. Navigate to this page using **Menu** > **Tools** > **Batch Run Tree**.

Description of Page

Select a **Batch Control** process and **Batch Number** to view information and statistics on the batch run's "threads". The following points should help understand this concept:

- Many batch jobs cannot take advantage of your hardware's processing power when they are run singularly. Rather, you'll find that a large percentage of the CPU and/or disk drives are idle.
- In order to minimize the amount of idle time (and increase the throughput of your batch processes), we allow you to set up your batch processes so that multiple instances of a given batch job are executed at the same time. For example, in Oracle Utilities Customer Care and Billing when you schedule the **billing** process, you can indicate that multiple parallel instances should be executed (rather than just one instance). You'd do this so that the processing burden of creating bills for your customers can be spread over multiple processes.
- We refer to each parallel execution of a batch process as a "thread".
- Statistics and information messages are displayed in respect of each thread. Why? Because each thread is a separate execution and therefore can start and end at different times.

The **Start Date/Time** and **End Date/Time** of the batch run are shown.

The tree includes a node that displays the total number of records processed for the batch run, the total number of records in error for the batch run and the batch run elapsed time. The elapsed time is the longest elapsed time among the batch thread(s). The message is red if there are any records in error. If the application properties file has been configured with the location of the log files and the log files associated with the batch run are still available, the links **Download stdout** and (if applicable) **Download stderr** are visible. Clicking either link allows you to view or save the log files. Note that the logs here are only related to the processing that occurs before the thread execution.

Note: Compression. The log files are compressed and include a *.gz extension. Different browsers treat this type of file differently. Some browsers may automatically decompress the file as part of the download and they are viewable in any text viewer by changing the extension to one the text viewer recognizes. However, some browsers download the compressed file and a user will need to unzip the file prior to viewing.

Note: Security Access. The 'download' hyperlinks are only visible for users that have security access to the **Download** access mode for the batch run tree application service.

Within each thread, a node appears for each batch instance for the thread. A thread will only have more than one instance in the case of a thread that ended in error where a user subsequently restarted the run. The **Start Date/Time** and **End Date/Time** of the batch run are shown, if populated.

The messages that appear under a thread instance show the start and end times of the execution instance. If errors are detected during the execution of the thread, these error messages may also appear in the tree. Refer to Processing Errors for information about the types of errors that appear in the batch run tree.

If the background process has been enabled to create To Do entries for object specific errors, information about the To Do entries are displayed in the tree. This information is not displayed for each thread, but rather all the To Do entries created for the batch run are grouped together. The To Do entries are grouped by their status.

If the log files associated with the batch thread are still available, the links **Download stdout** and (if applicable) **Download stderr** are visible. Clicking either link allows you to view or save the log files. The above notes about file compression and security access for downloads applies to these files as well.

Copyright © 2024, Oracle and/or its affiliates.

Batch Run Tree - Run Control

By default, if a batch process fails, it will restart. This tab allows you to modify the restart status of a failed run.

Navigate to this page using **Menu** > **Tools** > **Batch Run Tree** search for the desired batch control and then navigate to the **Run Control** page.

Description of Page

On the main page, you must select a **Batch Control**, **Batch Number**, and **Batch Rerun Number** to view a tree of the batch run. On this page, the following information is displayed:

- **Last Update Timestamp** contains the date and time the most recent batch run started or completed.
- **Batch Business Date** is the business date that was supplied to the background process (this date is used as the "system date" by the process).

Run Status indicates the status of the batch run. Valid values are: **In Progress**, **Error**, and **Complete**.

If the **Run Status** is **Error**, the system will attempt to restart this run when you attempt to execute the **Batch Control**. In most situations, this is exactly what you want to happen. However, there are rare situations where you do not want the system to execute a given batch run (e.g., if this run is somehow corrupt and you cannot correct the data for whatever reasons). If you want the system to skip the execution of a batch run (and proceed to the next run), turn on **Do Not Attempt Restart**.

Batch Day Tracking

The batch day tracking pages provide information to quickly determine the status and performance of the daily batch workload for a selected business date. The portals contain zones to provide details about run status and individual thread performance to assist in detecting data and performance issues with individual batch runs.

Batch Day Dashboard

This portal provides a high level summary of the status, duration and other details of the batch jobs that ran on a given business date. An implementation may introduce additional zones to this portal as needed.

Use **Menu** > **Dashboards** > **Batch Day Dashboard** to navigate to the portal.

Batch Runs By Business Day

This zone provides a summary by business date of the number of batch runs completed, in error, and in progress, together with the earliest start time and latest end time of the jobs on that day. The zone is sorted in descending order by business date.

Note: For performance reasons, the zone is configured to restrict the number of rows retrieved by the query. The 'Results' number will indicate how many rows have been retrieved. Use the Next and Previous buttons to retrieve additional dates or return to a prior set of dates.

The following filters are provided:

- **Batch Business Date** allows you to select a particular business date.
- **Process Date** allows you to restrict the batch runs included in the summary to those where the processing date is within the start and end dates for the run.
- **Exclude Reruns** allows you to exclude batch reruns from the summary.

Note: The values selected in these filters will be applied as hidden filters to the other zones in the portal.

When a specific business date is broadcast from the summary, the following zones will appear as part of the portal's **Main** tab page.

Copyright © 2024, Oracle and/or its affiliates.

Batch Day By Status

This zone summarizes the batch runs for the business date by their status, displayed in the form of a pie chart. Selecting a specific slice of data on the chart broadcasts its value to the other zones on the portal allowing them to filter their data accordingly. Use the **Reset** action to reset the filter set by this zone.

Batch Day By Category

This zone summarizes the batch runs for the business date by their batch category, displayed in the form of a pie chart. Selecting a specific slice of data on the chart broadcasts its value to the other zones on the portal allowing them to filter their data accordingly. Use the **Reset** action to reset the filter set by this zone.

Batch Jobs Summary

This zone displays details of the individual batch runs for the broadcast business date. The zone is sorted by batch control, batch run number and batch rerun number.

Note: For performance reasons, the zone is configured to restrict the number of rows retrieved by the query. The 'Results' number will indicate how many rows have been retrieved. Use the Next and Previous buttons to retrieve additional rows or return to a prior set of rows.

The following filters are provided:

- **Batch Control** allows you to select a particular batch control to review. This filter supports wildcard searching.
- **From Start Date/Time** restricts the batch runs displayed to those where the start date time is on or after this date and time.
- **To Start Date/Time** restricts the batch runs displayed to those where the start date time is on or before this date and time.
- **From End Date/Time** restricts the batch runs displayed to those where the end date time is on or after this date and time.
- **To End Date/Time** restricts the batch runs displayed to those where the end date time is on or before this date and time.
- **Batch Control Type** restricts the batch runs displayed to the selected type.

The following columns are displayed initially:

- **Batch Control** is the batch control ID of the batch run.
- **Description** is the description of the batch control. Click the hyperlink to navigate to the Batch Run Threads portal.
- **Batch Run** is the run number of the batch job. Click the hyperlink to navigate to the Batch Run Tree.
- **Status** is the status of the batch run.
- **Start Date/Time** is the start time of the run.
- **End Date/Time** is the end time of the run. The current date and time may be displayed if the batch job is still in progress.
- **Duration** is the time taken for the run, formatted as Days:Hours:Minutes:Seconds.
- **Threads** is the total threads for the batch run.

Click the **Column Configuration** icon in the zone header to show the Column Configuration Area in which additional columns can be added to the results. The additional columns include the following:

- **Batch Category** is the category of the batch run.
- **Batch Control Type** is the type of the batch run.
- **Do Not Attempt Restart** indicates whether the run has been marked not to be restarted.
- **Rerun Nbr** is the rerun number of this batch job.

Copyright © 2024, Oracle and/or its affiliates.

Batch Run Threads

This portal provides details and analysis of the threads for a given batch run. It is accessed via a link from the batch jobs summary zone on the Batch Day Dashboard. An implementation may introduce additional zones to this portal as needed.

The following zones will appear as part of the portal's **Main** tab page

- **Thread Summary.** This zone displays details of the individual threads for the batch run in context. It can be filtered by thread status.
- **Threads by Status.** This zone displays a count of the number of threads by status for the selected batch run, in the form of a pie chart.
- **Threads by Volume.** This zone displays a count of the records processed and records in error for each thread in the selected batch run, in the form of a bar chart.
- **Threads by Elapsed Time.** This zone displays the elapsed time for each thread in the selected batch run, in the form of a horizontal bar chart.
- **Threads by Records Per Second.** This zone displays the number of records processed per second for each thread in the selected batch run, in the form of a horizontal bar chart.

The Big Picture of Requests

Requests enable an implementer to design an ad-hoc batch process using the configuration tools.

An example of such a process might be to send an email to a group of users that summarizes the To Do entries that are assigned to them. This is just one example. The request enables many types of diverse processing.

- Request Type Defines Parameters on page 427
- Previewing and Submitting a Request on page 427
- To Do Summary Email on page 428
- Defining a New Request on page 428
- Setting Up Request Types on page 428
- Maintaining Requests on page 429

Request Type Defines Parameters

For each type of process that your implementation wants, you must configure a request type to capture the appropriate parameters needed by the process.

Previewing and Submitting a Request

To submit a new request, go to **Menu** > **Tools** > **Request** in add mode. You must select the appropriate request type and then enter the desired parameter values, if applicable.

After entering the parameters, the following actions are possible:

- Click **Save** to submit the request.
- Click **Cancel** to cancel the request.
- Click **Preview** to see a sample of records that satisfy the selection criteria for this request. This information is displayed in a separate map. In addition, the map displays the total number of records that will be processed when the request is submitted. From this map you can click **Save** to submit the request, **Back** to adjust the parameters, or **Cancel** to cancel the request.

When a request is saved, the job is not immediately submitted for real time processing. The record is saved with the status **Pending** and a monitor process for this record's business object is responsible for transitioning the record to **Complete**.

As long as the record is still **Pending**, it may be edited to adjust the parameters. The preview logic described above may be repeated when editing a record.

Copyright © 2024, Oracle and/or its affiliates.

The actual work of the request, such as generating an email, is performed when transitioning to **Complete** (using an enter processing algorithm for the business object).

To Do Summary Email

The base product includes a sample request process that sends an email to users that have incomplete To Dos older than a specified number of days.

The following configuration tasks are required to use this process:

- Define an Outbound Message Type. The business object usually defined for the Outbound Message Type is **F1-EmailMessage.**
- Define an External System that contains the Outbound Message Type in one of its steps. In the configuration determine if the communication is through SOA, batch, or real-time processing method when sending the email notification. Refer to Outbound Messages for more information about configuration needed for the different processing methods.
- Create a Request Type that includes the Outbound Message Type and the corresponding External System.
- Create a Request for the created Request Type.

Defining a New Request

To design a new ad-hoc batch job that users can submit via Request, first create a new Request Type business object. The base product BO for request type, **F1-TodoSumEmailTyp**, may be used as a sample.

The business object for the request includes the functionality for selecting the records to process, displaying a preview map for the user to review, and for performing the actual processing. The base product BO for request, **F1-TodoSumEmailReq**, may be used as a sample. The following points highlight the important configuration details for this business object:

- Special BO options are available for request BOs to support the Preview Request functionality.
  - **Request Preview Service Script**. This script retrieves the information that is displayed when a user asks for a preview of a request.
  - **Request Preview Map.** This map displays the preview of a request.
- The enter algorithm plugged into the Complete state is responsible for selecting the records that satisfy the criteria and processing the records accordingly.

Setting Up Request Types

Use the Request Type portal to define the parameters to capture when submitting a request. Open this page using **Admin** > **General** > **Request Type.**

This topic describes the base-package zones that appear on the Request Type portal.

Request Type List. The Request Type List zone lists every request type. The following functions are available:

- Click a **broadcast** icon to open other zones that contain more information about the request type.
- Click **Add** in the zone's title bar to add a new request type.

Request Type. The Request Type zone contains display-only information about a request type. This zone appears when a request type has been broadcast from the Request Type List zone or if this portal is opened via a drill down from another page. The following functions are available:

- Click **Edit** to start a business process that updates the request type.
- Click **Delete** to start a business process that deletes the request type.
- Click **Deactivate** start a business process that deactivates the request type.
- Click **Duplicate** to start a business process that duplicates the request type.

Copyright © 2024, Oracle and/or its affiliates.

- State transition buttons are available to transition the request type to an appropriate next state.

Please see the zone's help text for information about this zone's fields.

Maintaining Requests

Use the Request transaction to view and maintain pending or historic requests.

Open this page using **Menu** > **Tools** > **Request** > **Search**. This topic describes the base-package portals and zones on this page.

Request Query. Use the query portal to search for an existing request. Once a request is selected, you are brought to the maintenance portal to view and maintain the selected record.

Request Portal. This portal appears when a request has been selected from the Request Query portal. The following base-package zones appear on this portal:

- **Actions.** This is a standard actions zone.
- **Request.** The Request zone contains display-only information about a request. Please see the zone's help text for information about this zone's fields.
- **Request Log.** This is a standard log zone.

Copyright © 2024, Oracle and/or its affiliates.

Chapter 8

Attachments

Topics:

- Attachment Overview
- Configuring Your System for Attachments
- Maintaining Attachments

Some implementations may require that attachments be available from the application. These attachments can be stored in the Attachment table and then linked to other objects if applicable.

Attachment Overview

The following topics provide additional information regarding attachment functionality.

Attachment Types

The system supports several different attachment content types, for example:

- PDF Document
- Excel Spreadsheet
- Jpeg Image
- Text Document

The attachment data itself may be text or binary. When storing the data in the application however, it is stored as text information only. As a result, the upload of an attachment that is a binary type requires a conversion prior to storing the data. When viewing the attachment, the data is converted again for display.

Each type of attachment is defined using an attachment business object. The business object includes configuration defining the supported file extensions, whether the data is binary or not and the content type that represents the type of data for the attachment.

Note: To view the attachment business objects provided with the base product, navigate using **Admin** > **System** > **Business Object** > **Search** and search for business objects related to the Maintenance Object for Attachments (**F1-ATCHMT**).

Owned Attachments

Attachments can be either ‘owned’ or ‘common’. An owned attachment is one that is related to a specific record. For example, the specific test results for a given device can be uploaded and linked to that device or to its test records. These types of attachments are typically uploaded and maintained via the object that owns it.

Common Attachments

Common attachments are ones that are uploaded independent of any transaction in the system. They can be used for general system or company information. Or they can be linked to more than one transaction. For example, instructions for performing a certain type of task can be uploaded as an attachment and linked to a task type where those instructions are relevant. These types of attachments are uploaded and maintained in the central Attachment portal. Objects that may refer to the attachments may link the attachments via characteristics or some other appropriate mechanism.

Emailing Attachments

The system supports a business service that may be used by system processing to send an email. The business service **F1-EmailService** supports receiving the IDs of one or more attachments as input parameters.

Refer to Sending Email for more information.

External Reference

Attachment records may be added from an external system in which case, the external system could have a reference ID to capture. The attachment allows for including an external reference ID when adding an attachment internally.

Malware Scan

The system provides a plug-in spot where you can configure a step to scan attachment for malware. If an algorithm is provided, it is called every time attachment content is added.

Copyright © 2024, Oracle and/or its affiliates.

Configuring Your System for Attachments

In order to link attachments to objects in the system, there may be some configuration or implementation required to support the link. It is possible that one or more objects in your product already support attachments out of the box. Consult the product documentation for the specific object for confirmation. For objects in the system that do not support attachments out of the box, the following sections provide some guidelines for enabling support for attachments. Contact product support for more information.

Supporting Common Attachments

The attachments themselves are created / uploaded using the attachment portal. Refer to Maintaining Attachments for more information.

If your implementation has a use case where one or more common attachments may be linked to an object (and the object does not already support this functionality), the object may need to be extended to capture the attachments.

- If the object includes a characteristic collection, this is a recommended way to capture attachments. A characteristic type should be defined for each type of attachment. The characteristic type should be a foreign key type and should reference the Attachment FK reference. The characteristic entity collection should include the object that the common attachment will be linked to.
- Most characteristic collections are sequence based characteristics and would support multiple entries for the same characteristic types, if multiple attachments are applicable.
- If the object to support the attachments is governed by a business object, the implementation must extend the business object to define one or more appropriate elements used to capture the attachments. If only one attachment of a certain type is allowed, a single flattened characteristic may be used. If multiple attachments of a certain type are allowed, the BO schema may define a “flattened list” exposing the sequence and the characteristic type.
- If the object is maintained on a “fixed page” with a generic characteristic collection, no additional configuration is needed to allow users to link attachments to that object.

Supporting Owned Attachments

When creating an attachment for a specific record, the attachment itself captures the information about the related record, namely its maintenance object code and its primary key. For these types of attachments, no configuration is needed on the related business object to capture the attachments, as was the case with common attachments.

However, it is recommended to configure the user interface of the related object so that the owned attachments can be viewed and maintained from that page. To do this, you may use the generic attachment zone provided by the product: **F1-ATTCHOWN**.

Note that your product may already have support for viewing and maintaining owned attachments on one or more if its base delivered portals.

Configuring Size Limit

The system allows for configuration of a size limit for uploading attachments. This is optional but the recommendation is to set this to a reasonable value so as to limit the ability to upload unreasonably big files. To configure this, navigate to Feature Configuration. Search for an existing feature configuration with the Feature Type **General System Configuration**. If one does not exist, create a feature configuration of this type. Select the **Maximum Attachment Size** option and define an appropriate value.

Defining a New Attachment Type

As mentioned, the product provides support for several content types. If your implementation needs to support attachments for a content type not currently supported, create a new business object copying the configuration of an existing attachment business object.

Configure the following option types for the BO:

Copyright © 2024, Oracle and/or its affiliates.

- **Binary** indicates whether the attachment data must be converted from binary format. Binary attachments are stored in the database as text, and are then converted back to the original format when retrieved.
- **Content Type** represents the browser's mime type of the attachment.
- **Supported File Extension** specifies the valid file extensions for the content type.

Once the business object is defined, it is ready for use.

Supporting File Name Exceptions

By default, the system prevents attachment file names from having special characters. If there is some reason that an implementation needs to define an attachment with special characters, there is a system property setting that may be configured to relax this validation. Contact your system administrator for support.

Enabling Malware Scan

The product provides a malware scan algorithm in the base product that uses ClamAV. ClamAV itself is not provided as part of the product. To use the base algorithm, your implementation needs to get ClamAV and define system properties for the host and port where ClamAV may be called. The algorithm, called **F1-ATCHM-WS** should be plugged into the installation options - algorithms collection using the system event **Attachment Malware Scan**

If your implementation wants to use a different algorithm for malware scanning, you may deliver your own algorithm and plugit into the installation algorithm collection.

Maintaining Attachments

This section describes the functionality supported for viewing and maintaining attachments.

Navigate using **Admin** > **General** > **Attachment**. You are brought to a query portal with options for searching for common attachments.

Once an attachment has been selected, you are brought to the maintenance portal to view and maintain the selected record.

Note: The base search options for the attachments query only support searching for common attachments. Owned attachments may also be viewed on the attachment maintenance portal, but a user may only drill into the attachment maintenance from the maintenance portal of the “owning” entity.

The Attachment zone provides basic information about an attachment, including the ability to upload the file and to view an uploaded file. Viewing an attachment will typically launch a download of the attachment file, following standard browser behavior.

Adding Attachments

Common attachments may be added from the attachments portal (or via the standard menu path). In addition, your product may support attachments associated with specific records (“entity owned attachments”) which may also provide the capability to add attachments.

In both cases, when adding an attachment, you are prompted for the file to upload. Once the file is chose, the system determines the appropriate business object to associate with the attachment based on the file extension. Typically one and only one business object is found at which point you are prompted to provide the Attachment Name. (Your specific product may also require additional information at this time). Fill in the details and save.

Please note the following:

- If no business object is found for the uploaded file’s file type, an error is issued. This type of file is not currently supported as an attachment.
- If multiple business objects are found, the user must choose the appropriate one. This should be rare.

Chapter 9

Monitoring and Analytics

Topics:

- Reporting Tool Integration
- Service Health Check
- Measuring Performance
- Capturing Statistics
- Creating Cube Views
- Product Use Metrics

This chapter describes various tools provided in the product to support system monitoring including tools related to reporting, monitoring and analytics.

Reporting Tool Integration

This section describes how to configure your third party reporting tool and how to define your reports in the system to enable users to submit reports online.

The Big Picture Of Reports

The topics in this section describe the approach for designing and defining your system reports. Note that the product includes an out-of-the-box integration with BI Publisher. However it is possible to use the reporting objects to integrate with a different third party tool.

Integration with BI Publisher

Your DBMS, your product, and BI Publisher can work together to produce reports. You may choose to use a different reporting tool, but this may not be a trivial effort. This section provides high-level information about some of the business requirements that are being solved with the reporting solution.

Multi-Language and Localization Support

The integration supports a multi-language implementation and supports different localization settings.:

- All labels, headings and messages are defined using field and message meta-data in the application, which support multiple languages.
- The appropriate font, size, and layout are based on the requested report and the user's language.
- Dates and numbers are formatted as per the user's display profile.
- Currency based numbers are formatted as per the currency definition from the product

Requesting Reports from The System

Although reports are rendered in your reporting tool, users must be able to request ad-hoc reports from within the system (assuming users have the appropriate security access).

- The prompts for the input parameters must be shown in the user's language
- Users should be able to use the standard search facilities to find parameter values
- Plug-ins can be optionally used to cross-validate input parameters
- Application security must authorize ad-hoc report requests

Overview of the Data - BI Publisher

The following diagram provides an overview of where data is stored for your reports for integration with BI Publisher.

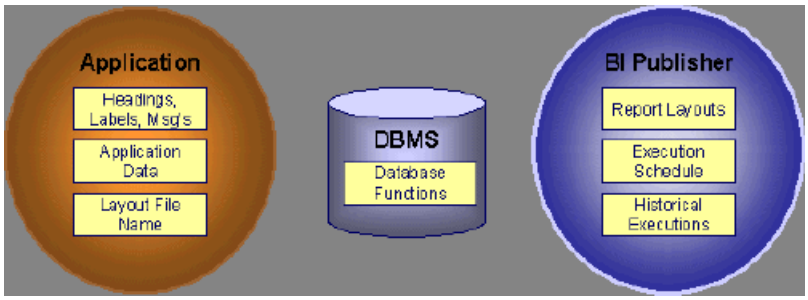


Figure 5: Application and BI Publisher

The application contains:

- The application data that appears on your reports.

Copyright © 2024, Oracle and/or its affiliates.

- The language-specific headings, labels and messages on your reports.
- The layout file name to be used for the report.

BI Publisher contains:

- How your reports look.
- Information about scheduled reports and reports that have already run.

The DBMS contains the SQL used to retrieve the data on your reports (residing in database functions).

Note: BI Publisher can be configured to retrieve data via a service call. Because every business object can be read via a service call, elements that reside in an XML based column can be displayed on reports produced using BI Publisher. See your product's *Installation Guide* or *Optional Products Installation Guide* for information on this configuration.

How To Request Reports

A user may request an ad hoc report from within your product:

- A report submission page enables a user to choose the desired report and enter the parameter values for the report
- The user must be granted security access to the report
- The request is passed to the reporting tool real time. Refer to Configure The System to Invoke BI Publisher for more information.
- The reporting tool generates the report and displays it in a new browser window

The reporting tools' scheduler creates reports (as per your schedule). This function is entirely within the reporting tool. No scheduling functions reside within your product.

A user can request an ad-hoc report from within the reporting tool. Note, the user's ID must be supplied as a parameter to the report in order for the user's profile to be used to format dates and numbers

Viewing Reports

As described above, ad-hoc reports requested from within your product are displayed immediately after they are generated in a new browser window

If your reporting tools supports it, the Report History page may be configured to open tool’s report execution history page and request a view of this report.

Note: The Report History page currently does not display historical reports for BI Publisher.

Configuring The System To Enable Reports

Configuring BI Publisher Reports

This section contains topics specific about configuring the product to interoperate with BI Publisher.

Configure the System to Invoke BI Publisher Real-time

The base product provides an installation algorithm plug-in spot called Reporting Tool. This plug-in spot should contain an algorithm that invokes the third party reporting tool real-time.

For BI Publisher, the system provides an algorithm type called **F1-BIPR-INV**, which invokes BI Publisher.

These algorithms rely on information defined in the Reporting Options table: the reporting server, reporting folder and the user name and password for accessing the reporting tool. The values in the reporting options should have been set up when the system was installed. Contact your system administrator if there are any problems with the values defined on the reporting options.

To use the algorithm types to invoke BI Publisher, perform the following steps:

Copyright © 2024, Oracle and/or its affiliates.

- Create an algorithm for the appropriate algorithm type.
- On the installation options, add an entry to the algorithm collection with an algorithm entity of **Reporting Tool** and indicate the algorithm created in the previous step.

Batch Scheduling in BI Publisher

For many of your reports, you probably want the report to be produced on a regular basis according to a scheduler. The reporting solution relies on the BI Publisher software to provide the batch scheduler functionality. Refer to BI Publisher documentation for details about configuring the batch scheduler.

Defining Reporting Options

The reporting options are provided as a mechanism for defining information needed by your reporting solution. The base product uses the reporting options to define information needed to access the reporting tool from within the system using the algorithm defined on the installation option.

Navigate to this page using **Admin** > **Reporting** > **Reporting Options**.

Description of page

Reporting Folder defines the shared folder where reports are stored.

Reporting Server From Browser defines the URL of the web application where the reporting tool is installed. For example, using BI Publisher, the format is: `http://<BI Publisher Server>:<port>`.

> **Note: Customize Options**. The reporting options are customizable using the Lookup table. This field name is **RPT_OPT_FLG**. The reporting options provided with the system are needed to invoke the reporting tool. If your implementation requires other information to be defined as reporting options, use the lookup table to define additional values for the reporting option flag.

Where Used

This information is used by the reporting tool algorithm on the installation option to invoke the reporting tool software.

Implementations may use reporting options to record other information needed for their reporting tool.

Defining Report Definitions

For each report supplied by your installation, use the report definition page to define various attributes of the report.

Report Definition - Main

Navigate to this page using **Admin** > **Reporting** > **Report Definition**.

> **CAUTION:** Important! If you introduce new report definitions, you must prefix the report code with **CM**. If you do not do this, there is a slight possibility that a future release of the application could introduce a new system report with the name you allocated.

Description of page

Enter an easily recognizable **Report Code** and **Description** for each report. Use the **External Reference ID** to define the identifier for this report in your external reporting tool.

Define an application service to enable users to request submission of this report online or to view report history for this report. Once you define an application service for each report, use Secured Objects and Application Services to define which users may access this report.

> **Note: Access Mode**. The access mode for application services related to reports must be set to **Submit/View Report**.

If you have more than one parameter defined for your report and you wish to perform cross-validation for more than one parameter, provide an appropriate **Validation Algorithm**.

Copyright © 2024, Oracle and/or its affiliates.

Enter a **Long Description** to more fully describe the functionality of this report. This information is displayed to the user when attempting to submit the report online or when viewing history for this report.

For BI Publisher, if you want to use one of the sample reports provided by the system, but with a different layout, indicate the layout to use for the report in the **Customer Specific Font/ Layout** field and BI Publisher uses this information instead. The name for base report layout is <report code>_Base. For example, a base layout for CM_TODO is named CM_TODO_Base.

Report Definition - Labels

Navigate to this page using **Admin** > **Reporting** > **Report Definition** and go to the **Labels** tab.

> **Note: Company name and logo.** Note the company name used as a title in the sample reports is defined as a message on the installation options. For information about installing the company logo, refer to your product's *Installation Guide* or the *Optional Products Installation Guide*.

Description of Page

In order to provide multi-language capability for each report, the labels used for the report must support multiple language definitions. For each label used by your report, indicate a unique **Sequence** and the Field used to define the **Label**. The label defined here should be the same label that is defined in your report layout defined in the external reporting tool.

When rendering an image of the report, the external reporting tool retrieves the appropriate label based on the language used for the report.

Report Definition - Parameters

Navigate to this page using **Admin** > **Reporting** > **Report Definition** and go to the **Parameters** tab **.**

Description of Page

The **Parameters** scroll contains one entry for every parameter defined for the report. The following fields display:

Parameter Code is the identifier of the parameter. This must correspond to the parameter definition in the reporting tool.

Required indicates that a value for the parameter must be defined when submitting the report.

Sort Sequence must match the parameter order defined in the reporting tool's report. It is also used when displaying the list of parameters on the report submission page.

Characteristic Type indicates the characteristic type used to define this parameter.

Default Value is option and if populated is displayed to the user when the report is chosen on the report submission page.

Description is a brief description of the parameter. This description is used when displaying the parameter on the report submission page.

Long Description is a detailed description of the parameter. This description is used on the report submission page when the user requests more information for a given parameter.

Sample Reports Supplied with the Product

Depending on your specific product, there may be sample reports provided that your organization may use as they are or as a starting point for creating a new report. The following sections provide an overview of the sample reports along with instructions on how to use one of the sample reports in your implementation environment.

How to Use a Sample Report Provided with the System

If you would like to use any of the sample reports, you need to perform some steps to be able to execute them in an implementation environment. This section walks you through the steps needed.

Copyright © 2024, Oracle and/or its affiliates.

Steps Performed at Installation Time

Refer to the *Installation Guide* or *Optional Products Installation Guide* for instructions for setting up and configuring your product and reporting tool to use the sample reports provided with the system. The following steps are described there.

- Setting up the stored procedures used by the sample reports.
- Defining the company title and logo used by the sample reports. Note the company name used as a title in the sample reports is defined as a message on the installation options.
- Defining a user for integration with your product.
- Publishing the sample reports in BI Publisher.

Contact your system administrator to verify that the above steps have occurred.

How To Define A New Report

Use a Sample Report as a Starting Point

- Make a copy of the report and save it in an appropriate directory. Prefix the new report name with **CM**.
- Review the stored procedure(s) used for this report. Refer to the installation guide for information about where the stored procedures should be defined. If you want to change the data that is being accessed, copy the stored procedure, prefixing the new stored procedure with **CM**. Make the appropriate changes in the new version of the stored procedure. Contact your database administrator to find out the procedure for creating a new stored procedure.

Note: Performance considerations. When designing a stored procedure, you must consider the performance of the report when executed. Consult your database administrator when designing your database access to ensure that all issues are considered.

Note: Defining Messages. The stored procedures provided with the system use messages defined in message category 30. If your new stored procedures require new messages, use message category 90000 or greater, which are reserved for implementations.

- Review the parameters used by the report. Make appropriate changes to the parameters required by the report. This affects how you define your report. Refer to Designing Parameters for more information.
- Determine whether or not you require cross validation for your report parameters. If any cross validation is necessary, you should design an appropriate validation algorithm to be executed when requesting a report in your product. Refer to Designing Validation Algorithms for more information.

Note: Cross Validation for On-line Submission Only. The cross validation algorithm is only executed for ad-hoc report submissions via your product. If you submit this report through your reporting tool, this algorithm is not executed.

- Review the labels used by the report. Labels and other verbiage are implemented in the sample reports using a reference to the field table in the system. This enables the report to be rendered in the appropriate language for the user. For any new report label you require, you must define a new field entry. Refer to Designing Labels for more information.
- Review the layout of the report and make any desired changes based on your business needs.

When you have finished designing and coding your new report in your reporting tool, you must do the following in order for it to be usable:

- Publish the report in BI Publisher. Refer to the documentation for this products for details about publishing a report. Refer to Publishing Reports in BI Publisher for configuration information specific to publishing a report for integration with your product.
- Define the report. Refer to Designing Your Report Definition for more information.

Copyright © 2024, Oracle and/or its affiliates.

Publishing Reports in BI Publisher

Please refer to the documentation for BI Publisher for more information about publishing a report in this system. The remaining topics in this section provide information about settings needed to ensure that the report is accessible using BI Publisher.

BI Publisher Database Access

When publishing a report in BI Publisher, you are asked for database logon information. The logon user name and password must be the user name and password that has access to the database functions related to this report in your database.

Verify BI Publisher User Access Rights

To verify the user's access rights to folders in BI Publisher:

- Open the BI Publisher Enterprise Security Center.
- Check that the role for the user has access to the appropriate report folders.

For more information, refer to the "Understanding Users and Roles" section in the Oracle Business Intelligence Publisher User's Guide.

Designing Your Report Definition

When adding a new report, you must define it in the system to allow users to request ad-hoc reports from on-line and to take advantage of the multi-language provisions in the system. The following topics illustrate the steps to take to correctly configure your report definition.

Designing Main Report Definition Values

Refer to field description section of the report definition main page for information about defining general information about the report.

For the validation algorithm, preliminary steps are required. Refer to Designing Validation Algorithms for more information.

For the application service, preliminary steps are required. Refer to Designing Application Services for more information.

Designing Characteristic Types

The parameter tab on the report definition page uses Characteristic Type to define the report parameters. For each report parameter that you plan to use, you must define a characteristic type.

You do not need a unique characteristic type for each report parameter. For example, if Start Date and End Date are parameters your report, only one **Report Date** characteristic type needs to be defined. This characteristic type would be used on both date parameters.

Each characteristic type to be used as a report parameter must indicate a characteristic entity of **Report**.

To illustrate the characteristic type definitions, let's look at the sample report Tax Payables Analysis. It needs the following parameters: From Date, To Date, GL Account Type Characteristic Type and Account Type value.

> **Note: Account Type Parameters**. The tax payables report must find general ledger entries that have posted to a certain distribution code. In order to find the appropriate distribution code, the report expects each distribution code to be defined with a characteristic indicating its GL account type (for example, **Revenue**, **Asset**, etc.) The report needs to know the characteristic type used to define this entry.

To support the required parameters, the following characteristic types are needed.

Copyright © 2024, Oracle and/or its affiliates.

Char Type	Description	Type	Valid Values	Char Entities
CI_DATE	Date Parameter	Ad-hoc	(Uses validation algorithm to validate proper date entry)	Report
CI_CHTYP	Characteristic Type	FK Reference	**CHAR_TYP**	Report
CI_GLTY	GL Account Type	Pre-defined	A- Asset, E- Expense, LM- Liability/ miscellaneous, LT- Liability/taxes, R- Revenue	Distribution Code, Report

Highlights for some of the above settings:

- We have defined a characteristic type for defining a characteristic type. This is to allow the user to indicate which Char Type on the Distribution Code is used for the GL account type. This is an FK reference type of characteristic.
- The GL account type characteristic type is referenced on both the Distribution Code entity and the report entity.

Designing Parameters

Your report definition parameters collection must define a unique parameter entry for each parameter sent to the reporting tool. The sequence of your parameters must match the sequence defined in your reporting tool.

Continuing with the Tax Payables Analysis report as an example, let's look at the parameter definitions.

Parameter Code	Description	Char Type	Default Value
P_FROM_DT	From Date	CI_DATE	N/A
P_TO_DT	To Date	CI_DATE	N/A
P_CHAR_TYPE	Account Type Characteristic	CI_CHTYP	CI_GLTY
P_TAX_ACCTY_CHAR	Account Type Char Value for Tax Related GL Account	CI_GLTY	LT-Liability/taxes

Highlights for some of the above settings:

- The from date and to date parameters use the same characteristic type.
- The characteristic type parameter is defined with a default value pointing to the GL account type characteristic type.
- The GL account type parameter defines the liability/taxes account type as its default value.

Note: User Id. The sample reports provided by the system pass the user id as the first parameter passed to the reporting tool. It does not need to be defined in the parameter collection for the report.

Designing Validation Algorithms

When designing your report definition, determine if cross validation should occur for your collection of parameters. In the Tax Payables Analysis report, there are two date parameters. Each date parameter uses the characteristic type validation algorithm to ensure that a valid date is entered. However, perhaps additional validation is needed to ensure that the start date is prior to the end date. To do this, a validation algorithm must be designed and defined on the report definition.

The system provides a sample algorithm **RPTV-DT** that validates that two separate date parameters do not overlap. This algorithm should be used by the Tax Payables Analysis report.

If you identify additional validation algorithm, create a new algorithm type. Create an algorithm for that algorithm type with the appropriate parameter values. Plug in the new validation algorithm to the appropriate report definition.

Copyright © 2024, Oracle and/or its affiliates.

Designing Application Services

Application services are required in order to allow a user to submit a report on-line or to view history for a report. Define an application service for each report and define the user groups that should have submit/view access to this report.

Update report definition to reference this application service.

Designing Labels

The system supports the rendering of a report in the language of the user. In order to support a report in multiple languages, the verbiage used in the report must be defined in a table that supports multiple languages. Some examples of verbiage in a report include the title, the labels and column headings and text such as "End of Report".

The system uses the field table to define its labels.

> **Note: Report Definition.** This section assumes that your new report in the reporting tool has followed the standard followed in the sample reports and uses references to field names for all verbiage rather than hard-coding text in a single language.

For each label or other type of verbiage used by your report, define a field to store the text used for the verbiage.

- Navigate to the field page using **Admin** > **System** > **Field**.
- Enter a unique **Field Name**. This must correspond to the field name used in your report definition in the reporting tool and it must be prefixed with **CM**.
- Define the **Owner** as **Customer Modification**.
- Define the **Data Type** as **Character**.
- **Precision** is a required field, but is not applicable for your report fields. Enter any value here.
- Use the **Description** to define the text that should appear on the report.
- Check the **Work Field** switch. This indicates to the system that the field does not represent a field in the database.

Update the report definition to define the fields applicable for this report in the **Labels** tab.

If your installation supports multiple languages, you must define the description applicable for each supported language.

Service Health Check

The system provides a plug-in spot on Installation, called Health Check. The algorithms plugged in here are used to check various system conditions and return details to help assess the health of the system.

What Information is Checked?

Each algorithm is responsible for checking one type of health check condition. The condition should be defined as a valid health component type (as defined in the **HEALTH_COMP_FLG** lookup). The algorithm may check conditions across many records (referred to as Health Components) and return a status and details for each health component. For example, if the algorithm is checking the level of service for batch programs in the system, the batch control is the health component in this case. The algorithm should gather the level of service results for each batch program and return the collection of information to the calling program.

The algorithms for the plug-in spot should return the following information for each health component.

- The **Health Component Detail** should be populated with information specific to the object that was checked. For example, if the algorithm is checking batch control level of service, this will have the Batch Control code. In addition, for enabling display for details of each health component type, the algorithm should populate the maintenance object code and primary key field and value details along with the navigation option.
- The **Health Component Status Flag** and **Health Component Status Description** should be populated with a valid status code and description appropriate for the condition being checked. This information could differ based on the health component type as each health component type may have its own unique status values.
- The **Health Component Status Reason** is meant to provide supporting information about the health component's status value. For example, if the component provides an error type of status, this could have the status reason.

Copyright © 2024, Oracle and/or its affiliates.

- The **Health Component Response** is used to map the status value of the health component to a standard value defined in the lookup **HEALTH_RESPONSE_FLG**. The values are a subset of HTTP response codes. The supported values are **All Checks Successful** (200); **Non-Critical Function Degraded** (203), **No Content** (204) and **One or More Critical Functions Degraded** (500).
- The **Response Details** list is provided for health components that check several conditions as part of the health check. Each condition could return its own status. In this situation, the algorithm should use the response details list to record the individual responses and populate the overall Health Component Status Flag, Description and Reason with summary information. The suggestion is to do the following:
  - Set the status flag and response based on the details of the individual responses. Refer to the batch level of service information below for an example of this logic.
  - When the response list is populated, the health check user interface will show the overall Status Reason text along with an icon to expand the details. The suggestion is to populate this field with text like "See Results for Details".

The system provides an algorithm that checks the **Batch Level of Service** health component type. This health component type finds all the batch controls that are configured with at least one level of service algorithm and invokes the algorithms for each batch control. The business service populates the output for this health service for each batch control as follows:

- The **Health Component Detail** is populated with the Batch Control code and description. In addition, the navigation information for being able to drill into the batch control are provided and used to build the column as hypertext.
- The **Status** is populated based on whether the batch control has one algorithm or multiple. If there is one algorithm, the Level of Service lookup value returned by the algorithm is returned. If there are multiple, the system determines an overall status based on the detailed status values from each algorithm. If any algorithm returns Error, that value is returned. Otherwise, if any return a Warning, that value is returned. Otherwise Normal is returned.
- The **Status Reason** is populated based on whether the batch control has one algorithm or multiple. If there is one algorithm, the expanded text of the status reason returned by the algorithm is returned. If there are multiple, the text from message category / message number 11002 / 22001 is returned (**See Results for Details**)
- The **Response** is populated based on the value of the overall Level of Service status. It is set to **All Checks Successful** (200) when the Level of Service is **Normal** or **Disabled**; **Non-Critical Function Degraded** (203) when the Level of Service is **Warning** and **One or More Critical Functions Degraded** (500) when the Level of Service is **Error**.

How are the Algorithms Called?

These algorithms are called by a business service provided by the product **F1-HealthCheck**.

That service calculates an overall Health Response value based on all the details returned by all the algorithms. The values described above for the **HEALTH_RESPONSE_FLG** are the ones used.

Also note that the system provides Inbound Web Services for this business service for both SOAP and REST service calls allowing external systems to use a web service to retrieve this information.

Health Check Configuration

To enable your system to display health check information, navigate to the Installation Options - Algorithms collection and add an entry for the **Health Check** system event for each health check algorithm provided by the product that your product would like to include in your system health check assessment.

Viewing Health Check Results

The product provides a portal that allows a user to view the detailed results of the health check business service.

Navigate using **Admin** > **System** > **Health Check** to view this portal.

The zone on the portal displays the following:

Overall Response is the HTTP response code returned by the business service.

Copyright © 2024, Oracle and/or its affiliates.

The grid displays the detail of each individual component checked as part of the system health check. Refer to Service Health Check for more information about the details returned.

Introducing Health Check Conditions

If your implementation would like to check a condition for system health that is not already provided by your product, here are the steps required:

- Define a new value in the Health Component Type lookup (**HEALTH_COMP_FLG**).
- Provide a new algorithm type, plug-in script and algorithm for the Health Check system event that checks the condition that you wish to check. Refer to the Service Health Check topic for the information that each algorithm should be supplying for the condition. Note that if you follow the pattern of the system delivered Batch Control - Level of Service algorithm, your algorithm may be designed to check conditions across many records of the same type and gather all the results.
- Plug this new algorithm into the Installation Options - Algorithms collection using the **Health Check** system event.

Measuring Performance

Many implementations need the ability to track and view the performance of key system processes against a defined target level. The framework provides objects to allow an edge product or an implementation to calculate and display performance measures against a desired target for one or more use cases. The topics in this section provide information about what is supplied in the framework and guidelines for implementing a specific use case for batch processes. Your product may supply out of the box support for additional use cases. Refer to your product documentation for more information.

Understanding Performance Targets

The following are examples of use cases that would be well suited for tracking as performance targets:

- Track and view the duration of key batch processes, either individually or as a group, and how they relate to a defined target.
- Monitor used and free space on a database against critical levels.
- Check the performance of individual user interface zones against a defined performance expectation
- Compare the number of web service requests made to an application against a threshold where performance may be of concern.

Framework provides functionality to define and categorize performance targets and link them to objects such as business services, zones and portals. This supports the calculation and display of the metrics against desired results.

In addition, Framework supplies out of the box support for batch process performance targets. Individual edge applications may supply more specific functionality for other use cases, if applicable.

Ideally, users should have the ability to view these performance targets on a dashboard that groups related measures. Framework provides the necessary components to achieve this for batch process performance targets.

The following sections highlight functionality supported for performance targets in the framework. Refer to the edge application product documentation for more details of other supported use cases.

Performance Target Objects Overview

The setup of a performance target involves a unique combination of configuration data and processing logic that calculates and displays a specific measure.

The framework performance target functionality is supported by a combination of inter-related objects, as shown below. Some of these objects will be generic for use in all performance targets while some are specific to a functional area such as batch processes.

Copyright © 2024, Oracle and/or its affiliates.

- **Maintenance Objects** for capturing performance target types and performance target instances.
- **Extendable Lookups** to define performance target categories and performance target metrics.
- **Business Services** to calculate known metrics for a group of performance targets, such as for batch processes.
- **UI Maps** to interpret the performance calculation results and display them in charts that show the comparison to the target.
- A **Zone** that serves as a template for system duplication to create specific zones related to each performance target. The zone invokes a business service to perform calculations and display performance measures in the related UI map.
- **Business Objects** to capture configuration data for a specific performance target instance and its related objects.
- Functionality to create **Zone instances** for specific performance targets based on the associated template zone.

Each edge application will deliver the following to compliment the objects delivered by the framework:

- **Portals** to group related performance target zones.
- Specific entries for the performance target category **Extendable Lookup** .

The following sections describe the combined use of these objects for performance targets in more detail.

Calculating and Displaying Performance Targets

Performance targets are intended to be displayed in a portal using an explicit object zone. The zone parameters define both the business service used to calculate the performance metrics and the UI map that displays the results. While an individual performance target needs to reference a zone with a unique configuration that calculates a particular metric, those zones will be based on a template zone which defines the core parameters.

Framework provides a base Batch Performance Target Metric Template zone (**F1-PERFBA**) for batch process performance targets. Refer to this zone and its parameters for more information about the batch performance zone configuration and the related business service and UI map.

Performance Target Metrics and Metric Types

The framework supports two types of metrics for performance monitoring:

- **Value based metrics** are used to record results against a specific numeric target.
- **Time based metrics** are used to track the results against a specific date and time target.

The list of valid metrics for a given performance target category and its associated performance target types is maintained using an extendable lookup. Framework uses the base business object Batch Performance Target Metric (**F1-BatchPerfTargetMetric**) to define batch process metric values. Refer to this lookup for the supported batch process metrics.

Note: While users are not prevented from adding new values to the lookup, the list is not intended to be extendable as new values will not be recognized by the business service that performs the base batch performance calculation logic.

Your edge product or implementation may supply other extendable lookup business objects for additional performance measurement use cases, if applicable.

Performance Target Categories and Types

There are key configuration details required by all performance targets. These are defined on two related objects.

Performance Target Categories

Target categories define the template zone and security setting for a group of performance measures. The list of valid categories is maintained using an extendable lookup.

The framework product supplies the business object Performance Target Category (**F1-PerformanceTargetCategory**) for this functionality. Refer to the business object description and configuration for more information.

Performance Target Types

Target types define the related performance target business object and the display portal for a group of performance measures. In addition, a target type references a target category which defines the associated zone details.

Copyright © 2024, Oracle and/or its affiliates.

The framework product supplies the business object Performance Target Type (**F1-PerformanceTargetType**) for this functionality. Refer to the business object description and configuration for more information.

The framework does not deliver any standard type or category values for batch processing performance targets. Refer to your specific edge application products to verify if any standard values are delivered for the batch processes within your applications. Edge applications may also supply standard categories and types for additional performance target use cases.

Performance Targets Define Specific Metrics

Although the types of measures and the business services and UI maps that govern how they are calculated and displayed are defined using separate objects, a **performance target** record defines the additional configuration needed to measure a specific metric and compare the result to a desired value.

There are key configuration details required by all performance target instances. These include a reference to the metric being measured, the desired target value or time, the desired result for the target and the unique zone by which this performance target will be monitored.

Note: The performance target maintenance object has a direct foreign key link to the extendable lookup business object and value that define the performance metric. This is an unusual pattern as extendable lookup values are normally recorded only in an object's schema. The pattern has been adopted to allow the description of the metric to be displayed in the performance target maintenance portal.

A given **performance target type** may require additional details for its calculations. For example, the framework batch process performance target defines additional details to restrict the measurement to specific batch processes that have executed within a given time frame. These details are configured on the performance target business object. Refer to the embedded help text on the business objects supplied by the framework and your edge applications for more information.

The **performance target type** also defines the business object to use when creating the resulting performance target record.

Objects Linked to a Performance Target

Some performance measures such as batch process metrics, derive the data for the calculation from objects within the system. The performance target can be linked to one or more related objects to define the specific data sources included in that target metric. For example, when creating a batch process performance target, the framework supports linking specific batch codes to the performance target to indicate the group of batch processes that are included in the measure.

The performance target business object may be configured to allow only relevant objects types to be linked to a performance target record. Refer to the base Batch Performance Target business object (**F1-BatchPerformanceTarget**) for an example.

Creating Performance Target Zones

Once a performance target and its objects have been defined, a unique zone needs to be created to display and monitor the specific target.

The framework functionality for batch performance targets implements the zone creation via the related business object lifecycle. When a performance target is added, status enter plug-ins are responsible for generating the new zone using the template zone and prefix configured on the target category (or the override values) and adding the zone to the portal configured on the target type. When a performance target is inactivated, an enter plug in is responsible for removing the zone from the portal.

Note: While the template zone associated with the performance target category may be overridden, the zone generation algorithm makes certain assumptions about the zone type and parameters. In particular, the logic expects to configure a zone parameter that references the performance target code as input to the business service responsible for calculating the metrics.

Refer to the base Batch Performance Target business object (**F1-BatchPerformanceTarget**) for details of the lifecycle and the enter plug-ins responsible for performance target zone creation.

Copyright © 2024, Oracle and/or its affiliates.

Setting Up Performance Target Configuration

The following topics highlight the general configuration steps required to use performance target functionality. Your particular product or implementation may supply additional functionality to support specific use cases for performance targets. Refer to your product's documentation and the library of business objects supplied for performance target in your system for more information.

Performance Target Category Lookup

Refer to About Performance Targets for an overview of performance target functionality.

Each Performance Target Type is associated with a unique Performance Target Category. The categories are defined as an extendable lookup.

Navigate to the Extendable Lookup portal. Search for and select the **Performance Target Category** business object. Define values that are appropriate to the categories your implementation is assigning for the performance target types in use. In some cases, an edge application may already have delivered the appropriate performance target category for your use

Refer to the embedded help for more information about configuring this object.

Defining Performance Target Types

Refer to About Performance Targets for an overview of performance target functionality.

To maintain the performance target types applicable to your product or implementation, open **Admin** > **Reports** > **Performance Target Type.**

This is a standard All-in-One portal.

The information captured on the performance target type depends on the business objects supported by your product or implementation. Refer to the embedded help text for more information.

Maintaining Performance Targets

This section describes the functionality supported for viewing and maintaining performance targets.

Refer to About Performance Targets for an overview of performance target functionality.

Navigate using **Main** > **Reporting** > **Performance Target**. This is an All-in-One portal and includes the standard List and display zones for a performance target.

For information about the data defined on the performance target, refer to embedded help.

Capturing Statistics

The product provides objects to allow for an edge product or an implementation to periodically calculate and capture statistics for one or more use cases. The topics in this section provide information about what is supplied in the framework and guidelines for implementing specific use cases.

Understanding Statistics

There are two objects that work together to capture the statistics related to a given business use case:

- The statistics control record which defines configuration related to capturing statistics. It also defines a category that is used to group similar types of statistics together.
- The statistics snapshot is the record where the actual calculated statistics are captured.

Copyright © 2024, Oracle and/or its affiliates.

Statistics Control

The statistics control record defines whether or not statistics are automatically calculated and how frequently. In addition, you may control the retention policy of snapshot records. You may indicate that only the most recent snapshot is kept or all snapshot records are kept or that they are retained for a defined number of days.

The system provides a business object for statistics control (**F1-Statistics**) that is expected to be used for most use cases. Its lifecycle includes the support for periodic capturing of statistics snapshot records as well as on demand capture.

Statistics Snapshot

The statistics snapshot object is the object that is responsible for calculating and capturing the statistics details. A separate business object must be defined for each use case. The schema of the business object defines the details that are captured. In addition, it must include an algorithm as an enter plug-in on the **Complete** state that is responsible for capturing statistics.

The system provides a 'root' business object for statistics snapshot (**F1-SnapshotRoot**) which includes the lifecycle that statistics snapshot business objects should follow. No explicit statistics snapshot use case is provided by the Framework. Your specific product may supply some out of the box support for certain use cases, in which case specific statistics snapshot business objects are provided. Refer to your product specific documentation for more information.

Viewing / Reporting Statistics

The product provides the mechanism for defining statistics control and statistics snapshot records and viewing the data in basic format. If your product has provided support for specific use cases, there may also be additional portals used to display the statistics in a meaningful format. For example, the statistics snapshot may be capturing data that can be presented in graphical format. Additionally, if multiple historical statistics snapshot records are retained, a zone may be defined to graph changes over time.

Configuring Your System for Statistics

If your product provides statistics use cases that you are planning to implement, all that you need to do is configure the appropriate statistics control record and define the appropriate configuration for your business requirements. Refer to Defining and Monitoring Statistics for more information.

If your implementation has identified an additional use case where you would like to capture statistics, the following points highlight the steps needed to configure the system to support the use case.

- Determine whether an additional Statistics Category value is needed. This is captured on the statistics category record. Navigate to the Lookup page. Search for and select the **STAT_CATEGORY_FLG** field. Review the values and determine if additional values are needed.
- Define a new statistics snapshot BO. This should be a child BO of the base delivered BO (**F1-SnapshotRoot**). Its schema should define elements for the specific information that is captured by the statistics calculations. The schema should be designed in conjunction with an appropriate Enter Status algorithm for the **Complete** that calculates the statistics as appropriate for the business requirement.
- Once the snapshot business object is designed and implemented, configure the appropriate statistics control record.

Defining and Monitoring Statistics

Refer to Understanding Statistics for an overview of the statistics functionality.

To define and maintain statistics control records and view a list of the current snapshot records, open **Admin** > **Reporting** > **Statistics Control.**

This is an All-in-One portal and includes the standard List and display zones for a statistics control.

For information about the data defined on the statistics control, refer to embedded help.

Statistics Snapshot

Copyright © 2024, Oracle and/or its affiliates.

If there are any Statistics Snapshot records for the statistics control, the **Statistics Snapshot List** zone displays a list of the most recent records. A user may drill into any of the records to view more detail. You are brought to a maintenance portal where you may view details about the calculated statistics. The information captured will be unique to the particular use case. Refer to the embedded help for more information.

Creating Cube Views

Many implementations require the ability to extract and analyze complex sets of data in a way that is simple to visualize and supports business or organizational decision making. A common model for performing this type of analysis is a multi-dimensional array, often referred to as a data cube. Cube views allow data to be represented in a way that provides the user with multiple perspectives of the results.

Although called a "cube", a cube view can have two, three or a higher number of dimensions. Each dimension represents some attribute in the database and the cells in the data cube represent a measure or value. For example, a cell could contain a count of the number of times that attribute combination occurs in the database, or the minimum, maximum, sum or average value of data related to that attribute. Queries are performed on the cube to retrieve information.

The framework provides objects to support the definition of data cubes and a flexible means of viewing them. The topics in this section provide information about what is supplied in the framework to assist in implementing a cube view. Your product may also supply out of the box support for relevant use cases. Refer to your product documentation for more information.

Understanding Cube Viewer

The framework provides a cube viewer portal which is intended to be used for most implementations of cube views. It supports a number of common actions and data representations.

Note: The choice of appropriate menu for the Cube Viewer portal will differ from product to product. Refer to your product specific documentation for more information on how to access the viewer.

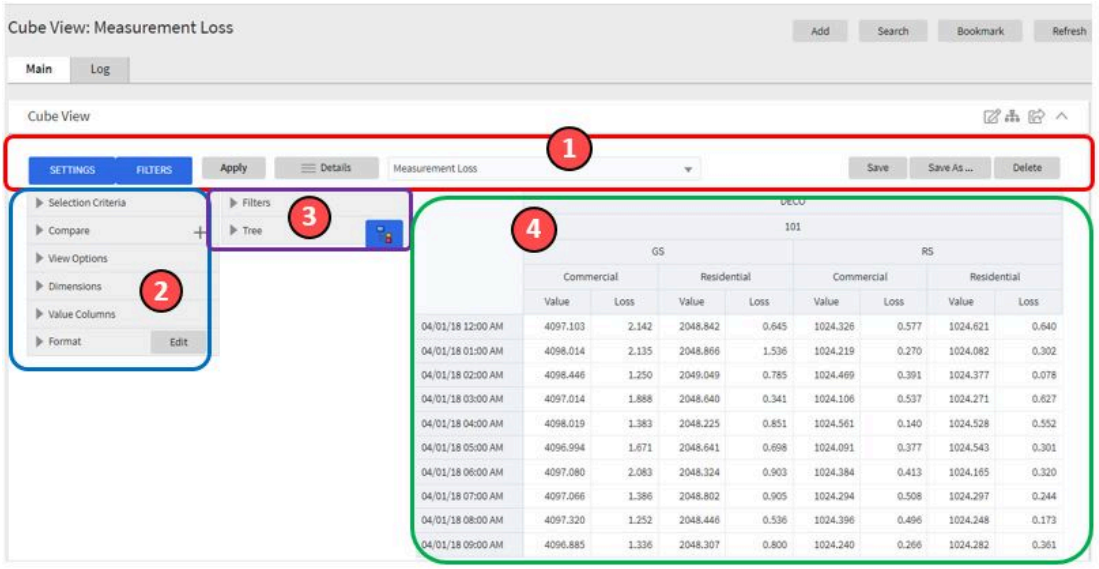
The following sections describe the basic concepts and features of the cube viewer.

Cube Viewer Components

The base Cube Viewer interface is comprised of the following main areas:

1. The Toolbar
2. The Settings Area
3. The Filters Area
4. The Views Area

Copyright © 2024, Oracle and/or its affiliates.



The next topics provide more information about each component.

The Toolbar

This following features are available on the Cube Viewer toolbar.

Settings and Filters Buttons

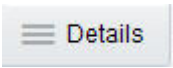
Click the **Settings** and **Filters** toggle buttons to collapse and expand the respective Settings or Filters areas.

Apply Button

Click the **Apply** button to apply any changes to the settings and display the new results in the views.

Details Button

Click the **Details** button to open a dialog box in which additional attributes of the current cube view can be viewed and updated. Refer to Cube View Details for more information about the attributes displayed.



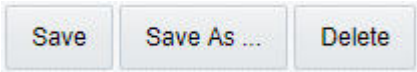
Cube View Dropdown

Click the **Cube View Dropdown** to open a list of cube views, of the same type as the current view, to which you have access. Selecting a cube view from the list will open that view. Use the search box to search for a view within the list using the cube view description.

Save, Save As and Delete Buttons

Click the **Save** button to save the current settings for the cube view. Click the **Save As** button to create a copy of the current cube view. Click the **Delete** button to delete the current cube view.

Copyright © 2024, Oracle and/or its affiliates.



Cube View Details

Clicking the **Details** button on the cube viewer **Toolbar** allows you to maintain the following information for the cube view:

- The short **Description** for display in the viewer header and a more **Detailed Description** for capturing additional information about the cube view.
- An **Access Type** of shared, private or public.
- The **Access Group** that defines the group of users with access to the cube view. This field is only visible if the access type is shared.
- A specific **User** who has sole access to the cube view. This field is only visible if the access type is private.
- The **Row Functions** text box in which the functions to be applied to the values across the rows in the data grid are defined. Clicking anywhere in the box will display a list of supported functions to choose from.
- The **Column Functions** text box in which the functions to be applied to the values in the columns in the data grid are defined. Clicking anywhere in the box will display a list of supported functions to choose from.
- An **Inactivate** button. Click this button to deactivate this cube view.

The **Fixed Parameters** for the cube view are displayed but cannot be updated here. Refer to Configuring Cube Types for more information on cube view parameters.

The Settings Area

The Settings Area contains a number of sections whose inputs control various aspects of the current cube view.

Selection Criteria

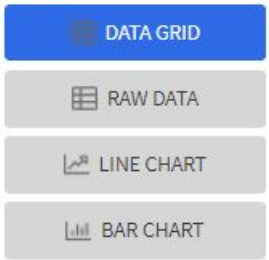
Open the **Selection Criteria** section to enter filter values for the user parameters defined on this cube type. User parameters that have been configured as 'Save with Cube View' on the cube type will be saved when the cube view is saved.

Compare

Open the **Compare** section to enter one or more sets of criteria that are used to retrieve data sets for comparison to the data retrieved by the main Selection Criteria. Use the Add and Delete buttons to add or remove a set of criteria. The comparison values will be displayed side by side with the main cube values in the data views. Note that the pseudo dimension **Dataset** must be included as a dimension column in order to display comparison views.

View Options

Open the **View Options** section to access buttons that toggle on and off the display of the supported data view types. The buttons can be used to change the order in which the views are displayed, using drag and drop. Any changes made to the default button settings will be saved when the cube view is saved. Refer to the Views Area topic for more information on the available data views.



Dimensions

Open the **Dimensions** section to view the list of dimensions defined on this cube type, as well as the pseudo dimension **Dataset** and the date/time filter, if applicable. Use drag and drop to move a dimension into the list of **Columns** or **Rows** displayed in the data views. Moving a dimension into the **Unassigned** list will remove it from the data views.

Copyright © 2024, Oracle and/or its affiliates.

Format

Click the Edit button on the **Format** section to open a dialog box in which conditional formatting, such as background or text color, can be defined for the values in the data grid. Refer to Defining Conditional Formatting for more information about the formatting controls.

> **Note:** Changes made to the settings will only be reflected in the views when the **Apply** button is pressed, with the exception of the **View Options**, where changes are reflected immediately upon toggling or rearranging the buttons.

Defining Conditional Formatting

Clicking the **Edit** button in the **Format** section of the **Settings** area allows you to define formatting rules for the display of values in the cube view data grid. The rules are based on mathematical expressions or formulas which use the cube view values as variables.

The formulas must be expressed using the field names of the values for the cube, as used in the cube view query zone. A list displaying the **Column Name** and associated **Label** for each value configured in the cube view is provided, to assist in defining the formulas.

The following attributes need to be defined for each rule:

- The **Sequence** defines the order in which the formatting rules are evaluated.
- The **Active** check box indicates whether the rule should be applied or not..
- The **Formula** defines the mathematical expression that is evaluated to determine whether the formatting should be applied. The expression variables should reference the **Column Name** of the value.
- **Format** defines the formats to be applied to cells whose value meets the condition expressed in the formula. The formats include the **Background** and **Text** colors and whether the text should be in **Bold**.
- The **Stop If True** check box indicates whether to stop evaluating the remaining formulas in sequence if this formula evaluates to true.

The Filters Area

The Filters Area provides the ability to select and apply dimension filters to the current view.

The **Tree** section displays a hierarchical view of the dimension values in the current cube view. Clicking a value will cause it to be displayed in the **Filters** section and immediately applies that filter to the data in the currently selected views. To remove a dimension from the filters, click on the value in the **Filters** section.

The **Hierarchical Filtering** icon, , is used to toggle between hierarchical and non-hierarchical selection mode. In hierarchical mode, clicking a dimension value at any level in the tree will automatically populate the filters with the related higher level dimension values. In non-hierarchical mode, only the selected dimension value is added as a filter.

Use the expand and collapse icons, , to quickly expand the tree, one level at a time, below the currently selected dimension value or collapse the entire level.

The Views Area

This section describes the types of views supported by the Cube Viewer.

Data Grid

The **Data Grid** displays the cube view data set in the form of a pivot table. The fields that appear as rows and columns in the table are defined in the **Dimensions** section of the Settings area. The default aggregation function used for the values is sum. Additional functions can be defined in the details of the cube view. Refer to the description of the **Details** pop-up window in theToolbar area for more information.

Raw Data

The **Raw Data** view displays the cube view data set as a flat table.

Line Chart

Copyright © 2024, Oracle and/or its affiliates.

The **Line Chart** displays cube view data in a chart with a time sequence as the X-axis and value intervals as the Y-axis. Each line corresponds to a specific combination of the selected dimensions. This view is only visible if the cube view includes at least one date/time filter in the selection criteria. If the criteria include more than one date/time field, the first date/time filter in order is chosen for the time sequence.

Bar Chart

The **Bar Chart** also displays cube view data in a chart with a time sequence and value intervals. Each bar corresponds to a specific combination of the selected dimensions. This view is only visible if the cube view includes at least one date/time filter in the selection criteria. If the criteria include more than one date/time field, the first date/time filter in order is chosen for the time sequence.

Cube Configuration Components

Cube views are supported using a number of standard configuration components.

The system provides a business object for cube views (**F1-CubeView**) which is used to capture access controls and the settings, such as user parameter values, for a cube view instance. The business object also defines the portal navigation and the base display UI Map (**F1-CubeViewDisp**) and accompanying display map service script (**F1_CubeViewD**). The display map and service script together drive the functionality supported in the cube viewer.

Note: The system also supplies a UI map fragment (**F1-CubeViewProc**) which is used to display a cube view in a process flow panel. Refer to Process Flows for more information.

The system provides a portal for cube views (**F1CUVWM**) that can be configured on the appropriate menu item or other navigation path for your implementation. The portal is linked to the base cube view zone (**F1-CUBEVIEW**).

In addition, the system supplies a simple query by cube view information (**F1-CUBVIWQ1**) that is the single entry on a base multi-query zone (**F1-CUBVIWQ**). The multi-query zone is not linked to the base portal as the expectation is that implementations will need to create searches tailored to their specific use cases for cube views. The base multi-query provides a quick means of configuring a cube view search while in development mode.

Configuring Cube Types

Cube Types are used to capture the details that define the sourcing of the data for a cube view and parameters that control certain features of the cube viewer. These details include:

- The business service whose associated zone defines the query used to select and filter the data for the cube view.
- The user parameters that correspond to the filters for this cube type's business service and zone.
- The user parameter settings that determine whether a given filter value should be saved with the cube view and whether the value should be populated from global or portal context, if applicable.
- The fixed parameter values to be applied as hidden filters for cube views of this type.
- The results from the business service and zone to be used as dimensions in the cube view and those that are the cube view values.
- The dimension settings that determine the order in which the dimensions are displayed in the data grid, which dimensions are to be used as filters in the cube view and which dimensions are to be included in the cube view tree.
- The values settings that determine the default sequence in which the values are displayed in the data grid.
- The FK Reference used to navigate from the data grid to the underlying source data.

If your product provides business services that support cube views you plan to use, all you need to do is configure the appropriate cube type records. Refer to Maintaining Cube Types for more information.

If your implementation has identified additional cube views you need for data analysis, you will need to create a zone and business service that use specific techniques for defining the SQL to derive the source data for the view and for mapping the results. Refer to Cube Type Advanced Topics for more information

Maintaining Cube Types

Refer to Creating Cube Views for an overview of the cube viewer functionality.

Copyright © 2024, Oracle and/or its affiliates.

To define and maintain cube type records, open **Admin** > **Analytics Configuration** > **Cube Type.**

This is an All-in-One portal and includes the standard list and display zones for a cube type.

For information about the settings defined on the cube type, refer to Configuring Cube Types and the embedded help.

Cube Type Advanced Topics

The following topics describe the technical models that need to be followed when creating a new cube type.

Designing Your Cube Type SQL

In order to support the base cube viewer, the SQL used in your cube type data explorer zone needs to be constructed with two main components. The first component creates a view of the data and its dimensions in a form that supports a tree-like structure and allows for the dynamic use of filters. The second component queries the filtered data set to provide the values to be displayed for each dimension.

The first component should contain the following queries:

- A sub-query constructed using a 'with' clause that builds a tree-like view of the cube's dimensions and optionally other columns used in the main SQL. This portion of the SQL must be enclosed within '/*>tree*/ ' and '/*<tree*/ ' hints and use the table alias **C_TREE**. The combination of the dimension columns should form a unique 'key' in the **C_TREE** data. Both the 'with' clause and the 'tree' portion within it may contain other supporting sub-queries.
- A sub-query which is the placeholder for filtered queries on the 'tree' data. It must use the table alias **C_FILTER** and take the form 'C_FILTER as (select * from C_TREE where 1=1 and ROWNUM <= n)'. The value of 'n' should be set to limit the rows returned to a number that is manageable within the cube viewer.

The second component is the main query that retrieves the measures for the cube view dimensions. This query references **C_FILTER** as the subset of the **C_TREE** and joins with other tables if required to retrieve the numeric values. The first date or date/time result column of the main query (as specified in the zone configuration) designates a time-series column, which forms the time-series 'key' with the dimensions columns.

Defining Your Cube Type Business Service

The business service that identifies the zone for a cube type is also used to provide additional configuration options.

The business service schema should define meaningful element names for the parameters, dimensions and values. It should also define labels for those elements to be used by both the cube type and cube viewer user interfaces. The labels can be derived by mapping the element to a meta-data fields or by using the '**label=**' syntax to define a text string directly in the schema definition. The '**required=true**' syntax can be used to require a value to be entered for a user parameter on the cube viewer or for a fixed parameter on the cube type.

The user parameter elements are expected be defined within a group in the schema called 'input' and the fixed parameters within a group called 'hidden'.

Product Use Metrics

The Product Use Metrics portal includes a zone (or zones) that are used to perform a business review of your organization's implementation of the product to confirm licensing / subscription compliance. The information visible on this portal will depend on the product or products installed by your organization.

To view this portal, open**Main** > **Dashboards** > **Product Use Metrics.**

For more information about what each zone on the portal is reporting, refer to the embedded help.

Copyright © 2024, Oracle and/or its affiliates.

Chapter 10

External Messages

Topics:

- Incoming Messages
- Outgoing Messages
- Web Service Category
- JMS Message Browser
- Oracle Integration Cloud Catalog
- XAI Documentation Note

This section describes mechanisms provided in the product that enable an implementation to configure the system to communicate with an external application.

Incoming Messages

This section provides information about support for incoming messages.

Inbound Web Services

Inbound web service functionality is provided to support receiving web service requests from an external system.

Understanding Inbound Web Services

The system supports communicating with the system via RESTful services or via SOAP services. In both cases, the system uses an object called inbound web service (IWS) to store the configuration. A web service class is used to distinguish whether the IWS is used for REST or SOAP. There is also a distinct business object for each web service class value because there is different configuration for each type of IWS.

For both types of inbound web service, the system supports the configuration of one or more operations per web service. Each operation defines the schema-based object to invoke to perform the desired function. An operation may refer to a Business Service, a Business Object, or a Service Script. If the IWS supports multiple operations, each operation can refer to the same or a completely different schema-based object from other operations within the IWS.

REST APIs may be secured by OAuth client credentials grant in a cloud installation. Refer to the Identity and Access Management chapter in the *Oracle Utilities Cloud Services Administration Guide* for more information.

The following topics provide more information about the different configuration of IWS based on whether it is SOAP or REST.

Inbound REST Web Services

This topic provides more information about the product's support of REST services using IWS.

The REST engine checks the Accept-Language header and attempts to find a supported language whose locale matches the referenced language. If found, the web service response returns any translatable data in that language. Note that if no supported language is found for the languages provided in the Accept-Language header, the system uses the language of the user that submitted the request.

IWS REST records are marked with a REST engine version of 1.0 or 2.0. Version 2.0 is the version used for new services. Version 1.0 is introduced for backward compatibility. Some of the system behavior differs based on the version of the REST service. The sections below highlight cases where this occurs.

HTTP Method and Parameters

When defining operations for inbound REST web services, the product supports the HTTP methods of Get, Patch, Post, Put, and Delete. Note that the product's support of these various HTTP methods are a means of communicating the purpose of the web service to the outside world. However, the actual behavior of the REST web service is driven by the behavior of the underlying schema based object (business object, business service or service script). For example, you may configure the HTTP Method of "Put" for an operation that references a service script that just retrieves a record. The product is not able to detect this type of discrepancy. Configuration users should carefully consider the correct method to use based on the logic of that service.

For operations that reference a business object, the transaction type must be provided. The REST syntax doesn't support defining the transaction type at runtime. There are some basic validation checks in this case related to the transaction type and the HTTP method. For example, a Get method only makes sense with the Read transaction type.

Note: Using the transaction type **Change** requires all values to be passed in. Using the transaction type **Update** allows the web service to pass only the primary key and the values to be updated. All other elements will retain their existing values.

Copyright © 2024, Oracle and/or its affiliates.

You may additionally define parameters. For each parameter, you define an external reference to the element, which is how this parameter is exposed to the external callers and is defined in the API specification. Each of these parameters is mapped to the XPath of the schema element from the underlying business object, business service or service script. For each parameter, you indicate if it is a **Path** parameter or a **Query** parameter.

- Path parameters are parameters that are part of the endpoint and are required. Each path parameter must be included in the operation's URI component surrounded by curly brackets.
- Query parameters are optional. They are not part of the endpoint but rather are included in the endpoint URL after a question mark, followed by name value pairs.

Refer to the URL section below for examples of path and query parameters in the sample URLs.

URL Composition

When building the endpoint URL for a REST service, there are three main parts that make up the full URL.

- The first part is the one that is environment specific. This will differ for an on premise implementation as compared to a cloud implementation. Both will have the host and the port and then additional components that identify the environment.
- The second part is a hard coded path designated by the product, namely "/rest/apis".

  These two parts of the URL are defined in the substitution variable **F1_REST_BASE_URL**. This is defined when initializing your environment. Refer to the Server Administration Guide for more information.

The remaining part of the URL is built dynamically based on configuration for each IWS and its operations. The components are "/ownerURIComponent/resourceCategoryURIComponent/iwsURIComponent/operationURIComponent"

- The owner URI component is taken from the owner flag of the inbound REST web service. A special extendable lookup (Owner Configuration for REST Services) defines this component for each owner flag.
- Each Inbound REST Web Service must reference a resource category. This category is used to associate related web services to a common category of resources. For multiple IWS records linked to the same resource category, external catalogues can use this information to group together related web services. The resource category is an extendable lookup and the URI component is an attribute of this record.
- Each REST IWS record defines a URI component that serves as an external identifier of this IWS record. The value must be unique within a given owner flag.
- Each operation must define the HTTP Method and optionally a URI component. When defining path parameters, the path parameters must be included in the URI component using curly braces.

In all cases, the URI component must begin with a slash ('/').

The following are some examples of the dynamic portion of the URL for a REST service. The last example illustrates the use of a query parameter.

IWS Name **URI Component**	**Owner** **URI Component**	**Resource Category** **URI Component**	**Operation HTTP Method** **URI Component**	**Dynamic URL Component**	**Run Time Example**
c1rateCalculation /rateCalculation	C1 /customer	C1-RATES /rates	Post /	../customer/rates/rateCalculation/	../customer/rates/rateCalculation/
w1WorkActivity /workActivity	W1 /asset	W1-WORK /work	Get /{activityId}	../asset/work/workActivity/{activityId}	../asset/work/workActivity/5798165498
w1WorkActivity /workActivity	W1 /asset	W1-WORK /work	Patch /scheduleWindow/{externalSystem}/	../asset/work/workActivity/scheduleWindow/{externalSystem}/	../asset/work/workActivity/scheduleWindow/{externalSystem}/

Copyright © 2024, Oracle and/or its affiliates.

IWS Name URI Component	Owner URI Component	Resource Category URI Component	Operation HTTP Method URI Component	Dynamic URL Component	Run Time Example
			{activityId}/{windowStartDateTime}	{activityId}/{windowStartDateTime}	MY-COMPANY/5798165498/20190
CM-AccountActivityHistory / accountActivityHistory	CM /cm	CM-ACCT-INFO / accountInformation	Get /{accountId}	../cm/accountInformation/accountActivityHistory/{accountId}	../cm/accountInformation/accountActivityHistory/1234567?activityId=5468976

Payload Format

REST services support receiving a request payload in XML or JSON format and returns the payload in either XML or JSON. The default format returned depends on the value of the REST Engine Version.

- Version 2.0 services assume JSON format as the default. The default can be overridden by providing an accept header of **application/XML**.
- Version 1.0 services assume XML format as the default. The default can be overridden by providing an accept header of **application/JSON**.

Root Node for JSON Format

Version 2.0 services with the JSON formatted payload do not accept in the request or return in the response any root node around the payload. The following is an example of the response of a REST call for a version 2.0 service:

```
{
   -"batchJobId": -"string",
   -"requestSuccessful": -"string"
}
```

Version 1.0 services with the JSON formatted payload expect a root node in the request and return one in the response. The following is an example of the response of a REST call for a version 1.0 service:

```
{
  -"F1CnclBatJob": {
    -"batchJobId": -"string",
    -"requestSuccessful": -"string"
 -}
}
```

The Open API Specification visible on the inbound web service maintenance page displays the expected format based on the record's REST engine version when viewing the specification.

Data Type Format in JSON

In JSON format, strings are surrounded by quotes and numbers and Boolean data do not have quotes. All services for REST engine version 2.0 follow this standard. Originally services that are REST engine version 1.0 were incorrectly treating numbers and Boolean data as strings and returning the data with quotes. This has been corrected.

To accommodate any integration that may have worked around the behavior for version 1.0 services, the system provides the ability to identify inbound web services that are exceptions to this rule. For any inbound web service identified as an exception, the system continues to treat numbers and Boolean data as strings in JSON responses. To add one or more IWS records to the exception list:

- Go to Feature Configuration.
- Look for an existing feature configuration with the feature type **External Messages**. If one does not exist, create one.
- Add an option for the option type **IWS JSON Data Type Exceptions**.

Copyright © 2024, Oracle and/or its affiliates.

- In the option value, indicate the IWS record that is an exception. Note that multiple options for the option type may be added. Additionally, the option value supports a comma delimited list.

External Facing Schema

By default the schema of the underlying service (for example, service script, business object or business service) is also the schema for the REST inbound web service operation and acts as both its Request and the Response schema. The product provides the ability to define an explicit schema for a REST inbound web service (IWS) operation, allowing a user to adjust the schema for an external facing consumer. The IWS operation schema also supports some special configuration that allows additional features to be defined only for the IWS operation schema.

The operation schema supports the following capabilities:

- Declaring whether an element belongs only to the request schema, only to the response schema, both (default) or excluded altogether. This is used to establish a clear separation between request and response schema definitions.
- Assign a different element name to an internal element. This allows the internal service element name to more closely align to internal references to the element, if needed. The external facing element name can be different allowing for a more readable schema.
- Introduce specialized elements that support links and other types of structural standards in a way that does not impact internal service processing.

Refer to Web Service Schema Nodes and Attributes for more information.

The REST servlet uses the request and response schemas to map elements to and from the internal schema.

Note: While the caller should only provide elements defined by the request schema, the application does not filter out extraneous elements. Assumption is that the internal service is designed to ignore such elements. However, only elements defined by the response schema are included in the response.

Dynamic Links

There are use cases where the published API includes a "_self" element that contain the endpoint URL of the GET operation related to the data returned in the response. Additionally, response payloads may include foreign keys and for those entities, the response includes a "_link" element that contains the endpoint URL of the GET operation for that entity (if it exists).

Syntax is provided in the REST inbound web service operation schema to support building the runtime endpoint URL for the _self and _link elements. Besides dynamically building the static portion of the URL based on the current environment details, it also builds the dynamic portion of the URL, substituting the URL components for the operation and substituting the path parameters. The syntax allows you to define a specific IWS Operation or allows you to reference a maintenance object and at runtime, the system determines the GET operation for the maintenance object and builds the URL accordingly.

Example of the syntax:

```
<_link getOperation="mo:'TO DO ENTRY';pk1:toDoEntryId;"/>
```

Example of the runtime endpoint URL:

```
_link: -"http://.../common/toDos/toDoEntries/28937296450934"
```

A GET operation may be associated with a maintenance object to denote that this is the default GET operation for the entity. A **GET Operation** option at the Maintenance Object and Business Object level may be used to override this default. If the entity is not associated with a business object or the latter is not associated with such option then the operation associated with the maintenance object (if any) is used.

Copyright © 2024, Oracle and/or its affiliates.

Open API Specification Documentation

The following documentation information may be provided for an Inbound Web Service:

- Short and detailed description for each operation.
- Help text may be provided for individual elements. The same text can be shared across multiple elements across all operations of the Inbound Web Service. If an element is not associated with help text then its internal field's label is used instead.
- Sample request and response documents may be provided for each operation.
- Operations may be associated with a sequence number that controls the order in which they appear in the Open API Specification.

Note: Detailed descriptions and element related help text are not translated since the published catalogue for base product APIs is in English language only.

Inbound SOAP Web Services

For inbound SOAP web services, by default the system uses the Schema Name to dictate the Request and Response for the service. The API can be overridden with custom formats by specifying Request and Response XSLs to transform into the relevant schema formats. In addition, if desired, the Request and Response Schemas that document the expected message may be provided.

Note: The Request and Response Schema fields are not supported in a cloud implementation.

Note: The system supports defining the XSLs as a managed content record. For backward compatibility, the system supports defining the XSL as a file in the file system. This is a system wide setting, defined using a feature configuration option. The feature type is **External Messages** and the option type is **XSL Location**. Set the value to **F1FL** to support XSL in the file system. The file location is configured using a system property. To support the XSL in managed content, no option is needed as this is the default. You may explicitly define the value of **F1MC** if desired.

In addition, note that for business object based operations, when invoking the web service an action is required. This may be passed into the web service as part of the invocation or alternatively, the action may be defined when configuring the operation using the transaction type.

Note: Using the transaction type **Change** requires all values to be passed in. Using the transaction type **Update** allows the web service to pass only the primary key and the values to be updated. All other elements will retain their existing values.

Annotations Used for Security

When preparing to deploy inbound SOAP web services, the security aspects of the service must be decided. The product provides a default security policy that is applied when no other policy is defined: **@Policy(uri="policy:Wssp1.2-2007-Https-BasicAuth.xml", attachToWsdl=true)** which requires HTTP Basic over SSL and a WS-Security Timestamp.

If a different security policy is desired, the following options are available:

- Security policies may be attached to the Inbound Web Service via the Java Enterprise Edition (Java EE) Web Application Server. This allows for multiple policies to be attached as supported by the Java EE Web Application Server. In order to enable this capability, explicit system configuration is required so that the product does not assume the default security policy. See the subsequent bullets for more information.
- Define a system wide security policy using a feature configuration option. Find the Feature Configuration record for the **External Messages** feature type. (It may need to be defined if it does not exist). Choose the option type **Default security policy** and define an appropriate value. If your implementation wishes for the policies to be attached at the Java EE Web Application Server, define this option type with an option value of **<none>**.
- Attach a security policy to the IWS via a Web Service Annotation. The base product provides annotation types that support the standard WS-Policy (**F1POLICY**) and OWSM Security Policy (**F1-OWSM**). No base annotation is supplied by the product for either annotation type.

Copyright © 2024, Oracle and/or its affiliates.

If your implementation wishes for the policy of a particular IWS to be attached at the Java EE Web Application Server, define a special annotation for the **F1POLICY** annotation type and configure the **uri** parameter value to **<none>**.

Note: Refer to WebLogic documentation for more information on supported security policies.

Note: In order to use the OWSM Policy, additional system configuration is necessary. Contact your system administrator to confirm if your implementation supports OWSM.

Inbound SOAP Web Service Deployment

A Inbound SOAP Web Service must be deployed to the Java EE Web Application Server in order for it to be available to the Web Service Clients to access the system. Refer to Deploying SOAP Web Services for more information.

Deploying XAI Inbound Service via IWS

For implementations using XAI inbound services for external messages, the product recommends moving to the inbound web service mechanism, which uses the Java EE Web Application Server to communicate with the product rather than the XAI servlet.

For XAI inbound services that use the **Business Adapter**, it is straight forward to move to IWS because the configuration is similar. In both cases, the service is configured to reference a business object, business service or service script. The associated WSDL for each record is similar. Changing the interface for the incoming message to use IWS instead of XAI inbound services is similar.

However, for XAI inbound services that use the **Core Adapter**, these services reference an underlying "page service" in the product. For these services, the Request and Response schemas for the XAI inbound service were created using the Schema Editor. In order to support calling an underlying "page service" in IWS, first a business service must be created to reference the page service (if one doesn't already exist). However, the resulting schema for the business service is different from the Request and Response schemas related to the XAI inbound service. Moving this functionality to IWS using business services requires changes to the format of the incoming messages.

Moving all incoming messages over to use IWS instead of XAI is the product recommendation. However, to aid in implementations that have many integrations in place using the XAI inbound services that use the **Core Adapter** (or any adapter whose message class is **BASEADA**), the product provides the ability to deploy these types of XAI inbound services to the Java EE Web Application Server along with the Inbound Web Services.

To take advantage of this capability, you must define a feature configuration option. Under the **External Messages** feature configuration type, the **Support XAI Services via IWS** is used to indicate if this feature is supported. Setting the value to **true** turns on the feature. If no option is defined for that option type, it is equivalent to setting the value to **false**.

When the system is configured to support XAI services via IWS, the Inbound SOAP Web Service deployment includes XAI inbound services (that are configured with an Adapter that references the **BASEADA** message class). The deployment portal will also include a zone showing the deployment status of these XAI Inbound Services.

Note: There is no support for XAI inbound services via REST, only via inbound SOAP web services.

Configuring Inbound Web Service Options

This topic describes the configuration needed for using inbound web services.

Configuring SOAP Inbound Web Service Options

This topics in this section describes the configuration needed for using inbound SOAP web services.

Technical Configuration

In order to use inbound web services, there are tasks a system administrator must perform.

Refer to the Server Administration Guide for technical details of each of these processes.

Copyright © 2024, Oracle and/or its affiliates.

Maintaining Web Service Annotation Types

The product provides some base annotation types. Refer to the metadata for more information. If your implementation wishes to define additional annotation types, use the Web Service Annotation Type portal. Open this page using **Admin** > **External Message** > **Web Service Annotation Type.**

You are taken to the query portal where you can search for an existing web service annotation type. Once an annotation type is selected, you are brought to the maintenance portal to view and maintain the selected record.

Note: Use of custom policies should only be considered if the policies supplied by the Java EE Web Application Server are not sufficient for your implementation's needs.

CAUTION: Important! When adding new records, carefully consider the naming convention of the web service annotation type code. Refer to System Data Naming Convention for more information.

The **Web Service Annotation Type** zone provides basic information about the web service annotation type.

Please see the zone's help text for information about this zone's fields.

The system supports the ability for an IWS record to refer to multiple policies. In this situation, the annotation type for the policy should include a reference to a parent annotation type so that the system can properly build the array of annotations.

Maintaining Web Service Annotations

If your implementation wishes to define annotations, use the Web Service Annotation portal. Open this page using **Admin** > **Integration** > **Web Service Annotation.**

This is a standard All-in-One portal.

CAUTION: Important! When adding new records, carefully consider the naming convention of the web service annotation code. Refer to System Data Naming Convention for more information.

Maintaining XAI Inbound Services

XAI Inbound Service is a legacy mechanism for exposing system services to external systems as web services. The current supported mechanism for defining web services is Inbound Web Services, which should be used for any new web service.

Some framework based products and existing implementations may have existing XAI inbound services that may need to be viewed. For these services, the product supports deploying them as SOAP inbound web services for execution. Refer to Deploying XAI Inbound Service via IWS.

The following sections describe basic information about the maintenance pages. Note that some of the information is related to legacy functionality that is no longer supported.

XAI Inbound Service - Main

To view an inbound service, open **Admin** > **XAI** > **XAI Inbound Service**.

Description of Page

XAI In Service Name is used in the system to identify the service. The service name is also the first XML element after the <Body> element in the XML request/response document. The **XAI Service ID** serves as the primary key.

Owner indicates if this XAI inbound service is owned by the base package or by your implementation (**Customer Modification**). This information is display-only.

The **Adapter** defines the interface with the target application server.

If adapter for this service invokes a system service, then the appropriate **Service Name** is visible.

If adapter is the **Business Adapter** then **Schema Type** and **Schema Name** reference the object to invoke.

Web Service Category is visible if the XAI inbound service is linked to one or more web service categories.

Copyright © 2024, Oracle and/or its affiliates.

Use the **Description** and **Long Description** to describe the service.

Check the **Active** switch if this service is enabled and available for execution.

Post Error is not applicable.

Trace is not applicable.

Debug is not applicable.

Schema Definitions

> **Note:** Request Schema and Response Schema are not applicable to services invoking schema-based objects. They do not appear when the **Business Adapter** is used.

The next two properties define the request and response XML schemas. The schemas are SOAP compatible. The schema XML files are expected to be stored in the Schemas Directory on the Web server running the XAI server.

The **Request Schema** is the XML schema defining the service request. The request sent to the server must adhere to the schema definition.

The **Response Schema** is the XML schema defining the service response. The response generated by the XAI server corresponds to the response schema definition.

The same service may perform several actions on a business object. Use the **Transaction Type** to define the default action performed by a service. The transaction type can be provided when invoking a service, by dynamically specifying a transaction type attribute on the Service element of the XML request. This field may take the following values: **Read**, **Add**, **Change**, **Update**, **Delete**, **List** and **Search**.

> **Note:** The difference between **Change** and **Update** is that for **Change**, all field values must be passed in with the request. Field values that are not passed in to the request are set to null. For **Update**, you need only pass the primary key field values and the values of the fields to be updated. All other fields retain their existing values.

Services, which perform a Search, may allow searching based on different criteria. When the Transaction Type value is **Search**, use the **Search Type** to define the default search criteria. The possible values are **Main**, **Alternate1**, **Alternate2**, **Alternate3**, **Alternate4**, **Alternate5** and **Alternate6**.

> **Note:** This is a default definition only and it may be overridden at run time when the service is invoked. To override the search type at run time, you should specify the searchType attribute on the Service element of the XML request.

XSL Transformation Definitions

Sometimes, the XML request document does not conform to the request schema, or the response document expected by the service requestor is not the one generated by the adapter. In such cases the request and/or the response documents must be transformed. The XAI server supports transformation through XSL transformation scripts. Transformation scripts may be applied to the request before it is passed to the adapter or applied to the response document before it is sent to the service requestor.

The **Request XSL** is the name of the XSL transformation to be applied to the request document before processing it. The transformation is usually required when the incoming document does not correspond to the XAI service request schema therefore it has to be transformed before it can be processed by the adapter.

The **Response XSL** is the name of the XSL transformation to be applied to the response document when the requester of the service expects the response to have a different XML document structure than the one defined by the response schema for the service.

Click the **WSDL URL** hyperlink to launch a separate window that contains the WSDL definition for the inbound service. Note that the server name and port number for the URL are built using a setting in the common properties file using the XAI HTTP Caller URL setting.

Configuring REST Inbound Web Service Options

This topics in this section describes the configuration needed for using inbound REST web services.

Copyright © 2024, Oracle and/or its affiliates.

Owner Configuration Lookup

Refer to Inbound REST Web Service for an overview of REST IWS functionality.

When generating the URL for a REST web service, part of the URL is taken from the owner of the Inbound REST Web Service record. The component to add to the URL is defined in this lookup using the URI Component field. The values are provided by the product.

To view the values, navigate to the Extendable Lookup portal. Search for and select the **Owner Configuration for REST Services** business object.

Resource Category Lookup

Refer to Inbound REST Web Service for an overview of REST IWS functionality.

Each Inbound REST Web Service is associated with a Resource Category. The categories are defined as an extendable lookup. The REST URL is built using a component from the IWS record's resource category.

Navigate to the Extendable Lookup portal. Search for and select the **Resource Category** business object. Define values as needed. Typically the category is related to a logical grouping of entities, or maintenance objects, referred to as a 'resource' in REST nomenclature. In some cases, an edge application may have already delivered appropriate resource categories for your use.

Populate the appropriate URI component to include when building the URL for IWS records linked to this resource category.

Maintaining Inbound Web Services

Inbound Web Services are used to define a specific message that your implementation will receive from an external system and provides configuration needed to process the inbound message.

The product provides support for REST and SOAP services and provides several inbound web services out of the box.

For inbound SOAP web services, by default no annotations are defined for the base inbound web services. You may modify the message options or the annotations for any base IWS record. In addition, you may define additional IWS records for other incoming messages supported by your implementation.

To view an inbound web service, navigate using **Admin** > **Integration** > **Inbound Web Service.** You are brought to a query portal with options for searching for inbound web services.

Once an inbound web service has been selected, you are brought to the maintenance portal to view and maintain the selected record.

CAUTION: Important! When adding new records, carefully consider the naming convention of the inbound web service code. Refer to System Data Naming Convention for more information.

The following zones may appear as part of the portal's **Main** tab page

- **Inbound Web Service.** This zone displays the configuration information for the record and its operations.
- **Operations.** This zone lists the operations defined for the selected inbound web service. The zone is only visible for a REST inbound web service. You may navigate to the Inbound Web Service Operation portal to further describe the operation's web service schema.
- **Help Text Fields.** This zone maintains help text information for fields across all operations of the selected inbound web service. The zone is only visible for a REST inbound web service.

Refer to Inbound SOAP Web Services for specific information about Operations for SOAP records.

Refer to Inbound REST Web Services for specific information about Operations for REST records.

Note that in addition to standard actions available on this portal, there is also a special **Add to Category** button in the page action area. Click this button to link an inbound web service to one or more web service categories.

Copyright © 2024, Oracle and/or its affiliates.

Maintaining Inbound Web Service Operations

This portal is used to maintain a REST inbound web service operation. You can access the portal from the **Operations** zone on the Inbound Web Service portal for a REST service.

Refer to Inbound REST Web Services for specific information about Operations for REST records.

The following zones may appear as part of the portal's **Main** tab page

- **IWS Operation.** This zone provides information about the selected operation.
- **Schema.** This zone describes the request and response schema as defined by the operation's web service schema. The zone is only visible when an operation schema is explicitly defined.
- **Schema Extensions.** This zone manages web service schema extensions that correspond to extensions made to the operation's internal schema. This zone is only visible if the underlying schema was extended using dynamic data area extension functionality. Each owner that extended the internal schema may provide web service schema information for their extension elements.
- **Options.** This zone lists various options for the operation such as request and response samples. The zone is visible when such options exist for the operation. Note that options are maintained as part of operation record.

The following zones may appear as part of the portal's **References** tab page

- **IWS Operation References.** This zone lists references to the current web service operation. The list includes other inbound web service operation records that reference the current operation via their web service schema. For a GET operation, the list also includes the maintenance object or business object that explicitly references the operation using a **GET Operation** option.

Web Service Schema Nodes and Attributes

A REST inbound web service (IWS) operation invokes a single schema based object, either a Business Object, a Business Service or a Service Script. The object has one schema which by default acts as both the Request and the Response schema. The product provides the ability to define an explicit schema for a web service operation that represents a view of the underlying internal schema for an external facing consumer. Note that the operation schema is a strict view of the internal schema and as such does not introduce structural changes.

While optional, defining an operation schema has the following benefits:

- Declaring the **usage** of an element as to whether it is part of the Request schema, Response schema, both or neither.
- Assigning an **external element name** to an internal element. Useful to avoid cloning of internal services for this purpose only.
- Better management and visibility of **GET operation** references.
- Support HATEOAS style structures outside of the internal schema. This allows for seamless internal processing.
- Providing help text for individual elements.

The following documentation is a complete list of the attributes available to you when constructing an IWS operation schema.

Operation Schema Definition

The following attributes defines the external facing operation schema that establishes the request and response schema for the operation.

Mnemonic	Valid Values	Description	Examples
usage=	**"REQ"**	Indicates that the element should be included in the request schema only. When specified for a group or list element the value applies to the entire container. A specific element	`<customerName - usage="REQ" -/>`

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Valid Values	Description	Examples
		in the container may override the container default.	
	"RESP"	Indicates that the element should be included in the response schema only. When specified for a group or list element the value applies to the entire container. A specific element in the container may override the container default.	`<customerId usage="RESP" -/>`
	"BOTH"	Indicates that the element should be included in both request and response schema definitions. This is the default value When specified for a group or list element the value applies to the entire container. A specific element in the container may override the container default.	`<customerName -usage="BOTH" -/>`
	"EXCL"	Indicates that the element should not be included in either the request or the response schema definitions. When specified for a group or list element the value applies to the entire container. A specific element in the container may override the container default.	`<httpMethod usage="EXCL" -/>`

mapTo=	"internal element name"	Maps a web service element to an internal element. By default the internal element name is assumed to be the same as the web service element name. Use this attribute to assign an external facing element name in the operation schema and map it to a corresponding internal element.
		Note that this is not an XPath expression but rather a reference to a valid element name that corresponds to the structure of the expanded internal schema.
		The owner of the element must be the same as the owner of the web service schema. If an operation's underlying schema was extended (using dynamic data area extension) then the extending owner may only reference their elements on their web
			Internal schema element:
			`<toDoPriority -.../>`
			Operation schema element:
			`<priority -mapTo="toDoPriority"/>`

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Valid Values	Description	Examples
		service schema extension for the operation.	

role=	**"FKGP"**	Establishes a group structure that represents a foreign key to an entity.
		
		The element itself does not exist in the internal schema. It groups together the foreign key value element(s) along with a corresponding **_link** element that at run time provides the endpoint URL of that entity's GET operation. Consumers of the web service may use the link to get the data for the entity.
			Internal schema element:
			
			`<user -.../>`
			
			Operation schema group:
			
			`<user role="FKGP">`
			`<user>`
			`<_link getOperation="mo: 'USER';pk1:user;"/>`
			`</user>`

	"COLL"	Establishes a group structure that represents a collection of entities that corresponds to an internal service list.
		
		The element itself does not exist in the internal schema. It groups together the internal list mapped to a **_data** element along with a corresponding **_link** element that at run time provides the endpoint URL for the collection's GET operation. Consumers of the web service may use the link to get a more detailed set of information for each entity in the list.
		
		By default all the elements in the referenced list are included in the operation schema. You may explicitly reference and define attributes for specific elements as needed.
			Internal schema element:
			
			`<alerts type="list">`
			`<alert/>`
			`<startDate -/>`
			`<endDate -/>`
			`<version -/>`
			`</_data>`
			`</alerts>`
			
			Operation schema group:
			
			`<alerts role="COLL">`
			`<_link - getOperation="... -"/>`
			`<_data mapTo="alerts">`
			`<version usage="EXCL"/>`
			`</_data>`
			`</alerts>`

getOperation=	"expression" Expression components: • **mo:**'<maintenance object>'; • **pk1:**<relative xpath to an element in the internal schema>; • **pk2:**...; • **pk3:**...; • **pk4:**...; • **pk5:**...;	Only valid on _self and _link elements. The attribute is used to reference the default GET operation for an entity as defined by its maintenance object or business object. The application determines the GET operation to use as follows: • If the entity's business object references an operation using the **GET Operation** option then it is used. • If the entity's maintenance object references an operation using the	`<_link getOperation="mo: 'TO DO ENTRY'; pk1:toDoEntryId;"/>`

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Valid Values	Description	Examples
	Parameters pk2–5 are required to match the maintenance object's prime key definition.	**GET Operation** option then it is used. • If a GET operation is found that references the maintenance object then it is used. If a GET operation cannot be determined for the entity a standard text indicating that a link is not available is returned instead.	
	"expression" Expression components: • **iws:**'<inbound web service name>'; • **operation:**'<operation name>'; • **parms:**[<external reference as defined on the GET operation record>:<relative xpath of an element name in the internal schema that holds the value>; <another external reference name>:<another relative xpath>;...];	Only valid on _self and _link elements. The attribute is used to reference a specific GET operation along with the information needed to compose its path parameters.	`<_self - getOperation="iws: 'F1-ToDoEntry'; operation:'getToDoEntry'; parms: [toDoEntryId:toDoEntryId;]"/ >`

Specialized Elements

The operation schema definition makes use of specialized element names in the style of the Hypermedia as the Engine of Application State (HATEOAS) standard. These elements should only be defined by the operation schema as they are not relevant to nor should be managed by the internal operation service.

The following list describes each element and how it is used.

Element Name	Description	Examples
<_self .../>	Response payloads may include a "_self" element that contains the endpoint URL of the GET operation related to the entity returned in the response. This is an optional element and only one may exist in an operation schema.	`<_self - getOperation="mo: 'TO DO ENTRY'; pk1:toDoEntryId;"/>`

Copyright © 2024, Oracle and/or its affiliates.

Element Name	Description	Examples
	Refer to **getOperation=** schema definition attributes for more information.	
<_link .../>	Response payloads may include foreign keys and for those entities, the response includes a "_link" element that contains the endpoint URL of the GET operation for that entity (if it exists). Refer to **role=** and **getOperation=** schema definition attribute for more information.	`<_link - getOperation="mo: 'TO DO ENTRY'; pk1:toDoEntryId;"/>`

<_data .../>	Provides a standard reference to a list of entities in the internal schema.
	
	Refer to **role="COLL"** schema definition attribute for more information.
		`<alerts role="COLL">`
		`<_link - getOperation="... -"/>`
		`<_data mapTo="alerts">`
		`</_data>`
		`</alerts>`

Documentation Attributes

The following attributes provide additional documentation for individual elements that is included as part of the web service's Open API specification.

Mnemonic	Valid Values	Description	Examples
helpText=	"help text field name"	Associates an element with a specific help text record defined for the Inbound Web Service. If the element is included in both request and response schema the same help text is used in both. If this attribute is not specified, and a single Help Text Field record exists for the Field associated with the internal element, that help text record is used. If more than one such Help Text Field record exists the one where the help text name is the same as the Field name is used. Note that an internal element is associated with a Field name via the mdField= attributes, if any, else via the mapField= attribute. No valid for _self and _link elements.	`<toDoEntryId usage="BOTH" helpText="TD_ENTRY"/>` In this example the inbound web service defines a help text field name "TD_ENTRY" that provides help text for this element. The same help text is included for the element in both request and response schema.
helpTextResponse=	"help text field name"	Associates an element that is included in both request and response schema with a specific help text record to be used in the context of the response schema.	`<toDoEntryId - usage="BOTH" - helpText= "TD_ENTRY" helpTextResponse= "TD_ENTRY_RESP"/>`

Copyright © 2024, Oracle and/or its affiliates.

Mnemonic	Valid Values	Description	Examples
		Follows same rules as the helpText= attribute.	In this example the inbound web service defines the following help text records: • Help text field name "TD_ENTRY" describes the element sent in the request schema. • Help text field name "TD_ENTRY_RESP" describes the element populated in the response schema.

Deploying Inbound SOAP Web Services

Once an Inbound SOAP Web Service is defined it is not automatically available to the Web Service Clients to access the system. The Deployment Status and the Active flag (set to true) indicate whether a Web Service is available or not. The last step is to deploy the Inbound SOAP Web Services to the Java EE Web Application Server. This deployment phase has a number of steps that are automatically performed when a deployment is initiated:

- The Web Service files are generated and policies are attached.
- The WSDL is generated with appropriate annotations and enumerations.
- The necessary Java stub code to implement the Web Service in the Java EE Web Application Server is generated and compiled.
- The Web Services are built into a valid Web Application Archive (WAR) file.
- Optionally, the newly created Web Services WAR file is deployed to the Java EE Web Application Server. This can also be done manually for clustered deployments, if desired.

There are two methods available for deploying inbound SOAP web services:

- Deployment at the command line using the **iwsdeploy[.sh]** command as outlined in the Server Administration Guide. This method is recommended for native installations and production implementations.
- Deployment using the Inbound Web Service Deployment portal. This method is only supported in development (non-production) environments.

Inbound Web Service Deployment Portal

To use the online Inbound Web Service Deployment portal, navigate using **Admin** > **Integration** > **Inbound Web Service Deployment**.

The following sections describe the base zones that are provided on the portal.

Deploy Inbound SOAP Web Services

The Deploy Inbound SOAP Web Services zone provides information about the last deployment. If the region is a development (non-production) region you may use the **Deploy** button to deploy or re-deploy inbound web services. All inbound web services whose Active switch is Yes will be deployed. All whose active switch is No will be undeployed.

Note: When an Inbound SOAP Web Service is deployed, the value of its service revision field is captured. Certain changes to configuration will require re-deployment to take effect. When any of the following changes occur, the IWS service revision value is incremented. This will cause the deployment status to show **Needs Deployment**.

- Active switch is changed
- An Annotation is added or removed
- An Operation is added or removed.
- The Operation Name, Schema Type / Schema Name, Request or Response Schema, Request or Response XSL for an Operation is changed.

Copyright © 2024, Oracle and/or its affiliates.

> **Note:** In addition, if the implementation supports XAI services deployed through IWS, the appropriate XAI inbound services will also be deployed or undeployed as required.

Deployment Status

The Deployment Status zone displays a list of inbound SOAP web services in the product, including the deployment status.

The deployment status is determined by comparing the internal Service Revision field on each IWS against the value captured at the time of deployment.

- **Deployed**. Indicates that the IWS has been deployed and no changes have been detected to the configuration.
- **Needs Deployment**. Indicates that the IWS has never been deployed or has been deployed but in the meantime, changes have been detected to the configuration that require redeployment.
- **Undeployed**. Indicates that the IWS is marked as inactive and the IWS is not found to be deployed at this time.
- **Needs Undeployment**. Indicates that the IWS is marked as inactive but the IWS is found to be deployed at this time.

If the IWS has been deployed, the View column will include a **WSDL** link allowing you to launch a separate window to view the WSDL definition.

> **Note:** For cloud installations, the system supports defining the WSDL URI via a substitution value defined for the token URI: @F1_BASE_IWS_URI@. The system will use this value per the functionality described in URI Substitution. When no value is defined, the system uses a default URL, which is the appropriate URL for on-premise installations.

Use the broadcast button adjacent to any of the inbound web services listed in the zone to view the details of the IWS record. This causes the **Inbound Web Service** zone to appear. It is the same zone that appears on the Inbound Web Service maintenance portal.

XAI Inbound Service Deployment Status

The XAI Inbound Service Deployment Status zone is only visible if the feature configuration option **Support XAI Services via IWS** is configured on the **External Messages** feature type or if the system detects that there are XAI inbound services that have been deployed. (The latter condition is checked for the case where an implementation has XAI inbound services deployed and then chooses to discontinue using this functionality. After changing the feature configuration option to false, one more deployment is required to "undeploy" the XAI services.) The zone displays a list of XAI inbound services in the product that are related to page services. Refer to Deploying XAI Inbound Service via IWS for more information.

The deployment status is determined by comparing the record's Version field against the value captured at the time of deployment.

- **Deployed**. Indicates that the XAI inbound service has been deployed and no changes have been detected to the configuration.
- **Needs Deployment**. Indicates that the XAI inbound service has not been deployed or has been deployed but in the meantime, changes have been detected to the configuration.
- **Undeployed**. Indicates that the XAI inbound service is marked as inactive or the **Support XAI Services via IWS** is not set to **true** and the XAI inbound service is not found to be deployed at this time.
- **Needs Undeployment**. Indicates that the XAI inbound service is marked as inactive or the **Support XAI Services via IWS** is not set to **true** but the XAI inbound service is found to be deployed at this time.

XAI inbound service does not have the equivalent of a Service Revision field that inbound web service has, which is only incremented when changes are made to the record that impact deployment. For XAI inbound service, the version number on the record is used. This field is incremented when any changes are made, even ones that may not impact deployment. As a result, some XAI Inbound Services may indicate "Needs Deployment" in cases where a redeployment may not be necessary. The recommendation when this occurs is to simply Deploy again to be safe.

If the IWS has been deployed, the View column will include a **WSDL** link allowing you to launch a separate window to view the WSDL definition.

Copyright © 2024, Oracle and/or its affiliates.

Guaranteed Delivery

There are alternatives for sending messages to the system besides using inbound web services. An external system may be able to send messages to the system in a generic manner where a new web service does not need to be defined for every new type of message. These types of messages may provide a payload (the message) and the service script or business service to invoke. An example of this type of communication is a message sent from a mobile application using RESTful operations.

The external system may have no mechanism for retrying failed messages. For this situation, the product provides an algorithm that may be used to capture incoming messages that should 'guarantee delivery'. A servlet processing this type of message may invoke the installation algorithm - Guaranteed Delivery, passing the details of the message and an indication if a response should be returned. The algorithm is responsible for storing the message information in a table so that it can be subsequently processed.

Outgoing Messages

"Outgoing messages" is the term used to describe messages that are initiated by our system and sent to an external system. Messages may be sent real time or near real time. The system provides the following mechanisms for communicating messages to external systems.

- **Outbound Messages.** This method allows implementers to use configurable business objects to define the message format and to use scripts to build the message. If sent near real-time the message is posted to the outbound message table waiting for Oracle Service Bus to poll the records, apply the XSL and route the message. If sent real time, the message dispatcher routes the message immediately.
- **Web Service Adapters.** Using a web service adapter, an implementation can consume a WSDL from an external system and create an "adapter" record that references the URL of the external system and creates appropriate request and response data areas to expose the payload information in a format understood by configuration tools. A script may then be written to build the request information and initiate a real-time web service call from within the system.
- **Send Email.** The system supplies a dedicated business service that may be used to send an email real-time from within the application.

All these methods are described in more detail in the following sections.

Outbound Messages

Outbound messages provide functionality for routing XML messages to an external system real-time or in near real time. In addition the functionality supports collecting related messages into a batch to then be sent to an external system as a consolidate XML message.

For each outbound message that your implementation must initiate you define a business object for the outbound message maintenance object. Using the business object's schema definition, you define the fields that make up the XML source field. These are the fields that make up the basis for the XML message (prior to any XSL transformation).

Each outbound message requires the definition of its schema by creating a business object whose schema describes the information that is provided to the external system. An XSL transformation may then be performed when routing the message to an external system.

For each external system that may receive this message, you configure the appropriate message XSL and routing information.

Because the outbound message type is associated with a business object, your implementation can easily create outbound message records from a script using the **Invoke business object** step type. Such a script would do the following:

- Determine the appropriate outbound message type and external system based on business rules
- Access the data needed to populate the message detail

Copyright © 2024, Oracle and/or its affiliates.

- Populate the fields in the schema and use the **Invoke business object** script step type for the outbound message type's business object to store the outbound message.
- The resulting outbound message ID is returned to the script and the scriptwriter may choose to design a subsequent step to store that ID as an audit on the record that initiated this message.

The following topics provide more information about functionality supported by outbound messages.

Polling Outbound Messages Using OSB

If the outbound message that needs to be sent to an external system can be sent as an asynchronous message (in 'near real time'), the process initiating the outbound message should create a record in the outbound message staging table. Oracle Service Bus (OSB), is the recommended tool to use to process outbound messages in near real-time.

Outbound messages that should be processed by OSB should be configured with a processing method defined as **SOA** for the external system / outbound message type. No other information is required for defining outbound message types that are processed by OSB.

For the OSB part of the processing, the product provides a custom transport: OUAF Outbound Message that may be used by an implementation to define messages to process and how to process them. This transport processes outbound messages in order of the priority defined on the outbound message type.

This section provides an overview of steps required to develop OSB integrations for outbound messages created by your product.

Before developing OSB integrations, a developer should be familiar with OSB development such as creating proxy services, business services, and message flow/routing. These terms are defined as follows:

Proxy Service: In OSB, a Proxy Service is the entity that processes a given type of message and routes it to a Business Service. A separate proxy service should be defined for each type of outbound message. If a given outbound message type may be routed to different external systems, it is the responsibility of the proxy service to query the external system defined on the outbound message and invoke the appropriate business service (see below). If any transformation is required prior to routing a message to a business service, it is the proxy service's responsibility to perform the transformation.

Business Service: In OSB, a Business Service is an entity that receives a message from OSB and routes it to the appropriate destination. This should not be confused with the Business Service object provided in the product in the configuration tools.

Fastpath: Refer to the technical reference paper *OSB Integration* for more information.

Batch Message Processing

Your implementation may be required to send messages to the same destination as a single XML file with multiple messages include. The following points describe this logic:

- The individual messages that should be grouped together must have a processing method of **batch** on the external system / outbound message type record. The appropriate batch code that is responsible for grouping the messages must also be provided.
- A separate "consolidated message" outbound message type should be configured for the external system with a processing method of **SOA**.
- When outbound message records are created for the individual messages, the batch code and current batch run number are stamped on the record.
- When the batch process runs it is responsible for building the XML file that is a collection of the individual messages. This batch process should include the following steps:
  - Format appropriate header information for the collection of messages
  - Apply the individual message XSL to each message before including the message
  - Insert a new outbound message for the external system with the "consolidated message" outbound message type.
- The consolidated message is ready to be processed by Oracle Service Bus.

Copyright © 2024, Oracle and/or its affiliates.

Note: No process provided. The system does not supply any sample batch job that does the above logic.

Real Time Messages

The system supports the ability to make web service calls, i.e. sending real time messages, to an external system.

The system supports special functionality for sending an Email message real-time. Refer to Sending Email for more information.

For other types of real-time messages, the system also uses outbound message type and external system configuration to format and route the message. When defining the configuration for real time messages, an additional step is required to define the mechanism for routing the message using a message sender. The system supports routing messages via HTTP and via JMS. Note that for HTTP routing, the system also supports sending the message using a JSON format.

Just like near real-time messages, initiating a real-time outbound message may also be done from a script. When a real time message is added, the system immediately routes it to the external system. If the external system provided a response message back, the system captures the response on the outbound message. If the outbound message type for the external system is associated with a response XSL it is applied to transform the response. In this case the system captures the raw response as well on the outbound message. Note that the outbound message BO should be configured to capture a response XML in its schema.

Any error (that can be trapped) causes the outbound message to be in a state of **Error**. It is the responsibility of the calling process to check upon the state of the outbound message and take a programmatic action. When the outbound message state is changed back to **Pending** the message will be retried.

The base package provides two business services: Outbound Message Dispatcher (**F1-OutmsgDispatcher**) and Outbound Message Mediator (**F1-OutmsgMediator**) that further facilitate making web service calls. Both business services are similar, allowing the calling script to configure the following behavior (with differences noted):

- Whether or not exceptions encountered while sending the message are trapped. Trapping errors allows the calling script to interrogate any errors encountered and take some other programmatic action.
- Whether or not the sent message is persisted as an actual outbound message record.
  - If a persisted message is desired, the recommendation is to use the Outbound Message Dispatcher. This business service creates the message using standard BO processing, relying on the outbound message logic to route the message and store the record. The message is routed after the BO pre-processing algorithm and after the record is persisted but before the BO post-processing and audit plug-ins are executed. If you need to send the outbound message ID as part of the message, refer to Define the Outbound Message Business Object and Type for more information.
  - If the message should not be persisted, then the recommendation is to use the Outbound Message Mediator. As mentioned, the Outbound Message Dispatcher creates the outbound message record, relying on the outbound message logic to route the message. If it should not be persisted, it is subsequently deleted. In contrast the Outbound Message Mediator executes the BO pre-processing algorithms and then routes the message explicitly without creating a message record. It is more efficient for scenarios that don't require persistence. Note that the Outbound Message Mediator also supports persistence, but it does so by creating the records without using BO processing. This is not recommended. The Dispatcher is the better option if persistence is desired.

Refer to the descriptions of the two business services for more information.

Designing the System for Outbound Messages

The following sections describe the setup required when using Outbound Messages to communicate with an external system. The configuration walks you through the steps to configure a single external system and all its messages.

Define the Outbound Message Business Object and Type

The product supplies many outbound message Business Objects along with Outbound Message Types for out of the box functionality.

In addition, implementations may need to define configuration for custom outbound messages. For each outbound message that must be sent to an external system, create a business object for the outbound message maintenance object.

Copyright © 2024, Oracle and/or its affiliates.

Using the business object's schema definition, your implementation defines the elements that make up the XML Source field (XML_SOURCE). These are the elements that are the basis for the XML message. XSL transformations may be applied to this XML source to produce the XML message.

If your integration is real-time and a response is expected, the Outbound Message business object should also map to the XML Response field (XML_RESPONSE).

- You may decide to capture the response as is and define the element as “raw”. For example

  ```
  <responseDetail mapXML="XML_RESPONSE" type="raw"/>
  ```

  In this scenario, a Response XSL may or may not be needed.
- Alternatively, if the details of the response are needed, you may define specific elements for the response. For this option, depending on how the integration is designed, a Response XSL may be needed to transform the response into the expected XML format.

Once you have your business object and schema, define an outbound message type for each unique outbound message.

Referencing an XSL

The system supports defining the XSLs as a managed content record. For backward compatibility, the system supports defining the XSL as a file in the file system. This is a system wide setting, defined using a feature configuration option. The feature type is **External Messages** and the option type is **XSL Location**. Set the value to **F1FL** to support XSL in the file system. The file location is configured using a system property. To support the XSL in managed content, no option is needed as this is the default. You may explicitly define the value of **F1MC** if desired.

Capturing the Outbound Message ID in the Message

If your integration would like to use the system generated Outbound Message ID as a unique identifier with the external system, the following configuration is needed:

- Define an element within the XML Source that should be populated with the system generated outbound message.
- Configure a BO Option on the outbound message’s business object using the Option Type **Outbound Message ID XPath** and set the Option Value to the XPath of the element defined to capture the ID.

Note: This functionality is only applicable if the outbound message is persisted.

Support for Dynamic URLs

The product supports the ability to build a dynamic URLs. These are cases where the URL requires information determined at runtime. This is supported with a combination of BO schema definition, URL configuration and appropriate code when creating the outbound message. The following points highlight the steps needed to support this functionality.

- When defining the Message Sender’s URL, use the syntax **${pathParms}** in the location of the URL where runtime information must be inserted. For example: **http://[SERVER_NAME]:1000/rest/services/${pathParms}**
- Include the data area **F1-OM-DynamicConfig** (Outbound message dynamic configuration) in the schema of the business object for the outbound message. This data area includes elements for **pathParms** and **queryParms**.
- In the code that creates the outbound message, populate the **pathParms** and if applicable **queryParms** elements with the appropriate information. The system will build the URL by plugging in the **pathParms** element value followed by a question mark, followed by the **queryParms** element value into the **${pathParms}** location in the URL.

The following is an example of run time values.

```
    <dynamicConfiguration>
        <pathParms>job/1234</pathParms>
        <queryParms>firstName=John&lastName=Doe</queryParms>
     <dynamicConfigureation>
```

Copyright © 2024, Oracle and/or its affiliates.

Add SOAP Header Parameters at Runtime

The product supports the ability to add SOAP header parameters to an external message at runtime. The following points highlight the steps needed to support this functionality.

- Include the data area **F1-OM-DynamicConfig** (Outbound message dynamic configuration) in the schema of the business object for the outbound message. This data area includes the element **soapHeaders** .
- In the code that creates the outbound message, populate the **soapHeaders** element with the self contained XML to add to the SOAP header section of the outgoing SOAP request.

Real-Time Message Configuration

Messages are routed to an external system real-time using the outbound message dispatcher or using the real-time send email business service. The system supports routing messages using HTTP and routing messages using JMS. In addition, there is a special type of message sender used for sending emails. The following sections highlights the supported real-time communication and the configuration needed for each.

Email Messages

For sending emails, the following configuration is needed:

- Define a message sender configured for email. Senders of this type should be configured with the **RTEMAILSNDR** class. The sender context is used to configure the connection information for the connecting to the SMTP server.
- This sender may be defined as the default email sender on the message option table. Alternatively, the message sender can be provided to the business service as input. Refer to Sending Email for more information.

Outbound Messages

For other outbound messages that are routed using the real-time outbound message dispatcher, a message sender must be configured to define how to route the message. The following points highlight more detail related to this configuration.

Determine the communication mechanism prior to configuring the sender.

- When routing the message using JMS, the following configuration must be defined
  - Define an appropriate JNDI Server that indicates where the JMS resources are located.
  - Define a JMS Connection to define additional configuration needed for the connection.
  - Define the JMS Queue or JMS Topic to define the queue or topic to use.
- When communicating using a JSON format, determine the method to use to convert the XML to JSON. The desired method is driven by how the request should be sent.
  - If choosing the **Base JSON Conversion** method, if XSL transformation needs to be applied prior to the conversion to JSON, then the target XML Request Schema must be defined (using a data area) so that the conversion logic knows the format of the XML it is converting. The XSL is applied to the outbound message's XML Source, resulting in the defined XML Request Schema, which is then converted to JSON. If XSL transformation is not required, then the outbound message's XML Source is converted to JSON.
  - If choosing the **Rootless JSON Conversion** method, the group element mapped to the XML source field is removed by the conversion, resulting in a rootless JSON request document.
  - If the XML source on the outbound message can be converted to JSON using an XSL, then the **XSL Transformation** method may be chosen.
  - You may also choose to convert the XML Source to JSON via the **Standard API Conversion** method (using a Jettison library). With this method, an XSL may optionally be provided. The conversion will be performed on the transformed XML.
  - For the response, if the outbound message BO defines detailed elements for the XML Response field, then the JSON should be converted to this format.
    - If your conversion method is **Base JSON Conversion**, then if the JSON response cannot be converted directly to the XML Response elements on the outbound message BO, then define a Response Schema (data area) that represents the results for the base JSON conversion. In addition, define an XSL that can transform the response from the converted XML to the XML format expected on the BO.

Copyright © 2024, Oracle and/or its affiliates.

    - If your conversion method is **Rootless JSON Conversion**, the response document is assumed to be rootless. The group element mapped to the XML response field is added by the conversion process, resulting in a well formed XML response document.
    - If the conversion method is **Standard API Conversion** or **XSL Transformation**, the standard API is used to convert JSON to XML. An XSL may be defined to convert the response to the XML Response if needed.
  - If the outbound message BO defines a "raw" element to capture the response, then a response schema and XSL are not necessary. In this case, the system will perform a JSON to XML conversion using the **Standard API Conversion** method (regardless of the conversion method defined) and the result is captured in the XML response.
- For HTTP senders including JSON senders, the system provides the following support for sending messages secured by OAuth authentication:
  - Using Oracle Web Services Manager (OWSM). The system provides a pre-configured set of policies for OAuth (**F1-OAUTH**) using a special extendable lookup. Note that the values of this policy set defines a specific CSF key repository that the implementation should use for capturing its CSF keys. In addition, there is a substitution value defined for the token URI: @F1_OAUTH2_URI@. Configure the appropriate URI for this implementation as described in URI Substitution. By default the system does not support additional policy sets to be defined. If your implementation requires a different policy set, contact support.
  - You may provide OAuth related settings as part of the message sender configuration for a REST API.

Define a message sender configured for each appropriate routing method. The invocation type should be configured as **Real-time**. For routing via HTTP, use the **RTHTTPSNDR** - HTTP sender class. For routing via HTTP with SOAP format automatically applied, use the **SOAPSNDR** - HTTP SOAP sender class. For routing via HTTP using JSON format, use the **RTJSONSNDR** - JSON sender class. For routing via JMS, use the **RTJMSQSNDR** - JMS queue sender class or **RTJMSTSNDR** - JMS topic sender class and configure the JMS Connection and JMS Queue or JMS Topic. Use the sender context to configure the required values for connecting to the appropriate destination.

Note: Refer to Support for Dynamic URLs for configuration needed to support dynamic URLs when sending an outbound message. There is specific configuration expected when defining the URL on the Sender in order to support dynamic URLs.

Configure the external system / outbound message type collection. The processing method defined for the external system / outbound message type must be **Real-time**.

Define the External System and Configure the Messages

Define an external system and configure the valid outgoing messages and their method of communication (processing method). Refer to Outbound Messages for more information.

Outbound Message Schema Validation

The outbound messages that are generated by the system should be well formed and valid so that they do not cause any issues in the external system. To ensure this validity you may configure the system to validate messages before they are routed to their destination. Note that the validation is applied just before communication with the sender and therefore after any Request XSL has been applied.

- Define a directory where the valid W3C schemas are located using the Message option **Outbound Message Schema Location**.
- Each external system message must indicate the appropriate W3C schema to use for validation.

You may turn on or off this validation checking using the Message option **Schema Validation Flag**.

This feature is not supported for cloud implementations.

Configuring the System for Outbound Messages

The following sections describe the setup required when using Outbound Messages to communicate with an external system.

Copyright © 2024, Oracle and/or its affiliates.

JNDI Server

If using JMS to communicate outbound messages, define a new JNDI Server. Open **Admin** > **Integration** > **JNDI Server**.

Description of Page

Enter a unique **JNDI Server** and **Description**.

Indicate the Provider URL to indicate the location of the JNDI server. A variable may be used in place of all or part of the URL. The variable must be predefined in the substitution variable property file. The value here should enclose the variable name with @. Refer to Referencing URIs for more information.

Indicate the **Initial Context Factory**, which is a Java class name used by the JNDI server provider to create JNDI context objects.

JMS Connection

To define a JMS Connection, open **Admin** > **Integration** > **JMS Connection**.

Description of Page

Enter a unique **JMS Connection** and **Description**.

Indicate the **JNDI Server** to be used. Refer to JNDI Server for more information.

Use the **JNDI Connection Factory** to indicate the lookup keyword in the JNDI server used to locate the JMS connection.

Where Used

JMS Queue

To define your JMS Queue values, open **Admin** > **Integration** > **JMS Queue**.

Description of Page

Enter a unique **JMS Queue** and **Description**.

Enter the **Queue Name** as defined in the JNDI server. This is the JNDI lookup name identifying the queue.

Use the **Target Client Flag** to indicate whether or not the target client is **JMS** or **MQ**.

Select the **JNDI Server** where the queue is defined. Refer to JNDI Server for more information.

JMS Topic

To define your JMS Topic values, open **Admin** > **Integration** > **JMS Topic**.

Description of Page

Enter a unique **JMS Topic** and **Description**.

Select the **JNDI Server** where the topic is defined. Refer to JNDI Server for more information.

Enter the **Topic Name** as defined in the JNDI server. This is the JNDI lookup name identifying the topic.

Message Sender

The topics in this section describe the maintenance of a message sender

Message Sender - Main

To define a new sender, open **Admin** > **Integration** > **Message Sender.**

Description of Page

Enter a unique **Message Sender** and **Description**.

CAUTION: Important! When adding new records, carefully consider the naming convention of the message sender code. Refer to System Data Naming Convention for more information.

Copyright © 2024, Oracle and/or its affiliates.

Set **Invocation Type** to **Real-time**.

Indicate the **Message Class** for this sender, which indicates the method used to route the message. The valid sender classes are as follows:

Message Class	Description
RTEMAILSNDR	Email sender.
RTHTTPSNDR	HTTP sender.
RTJMSQSNDR	JMS queue sender.
RTJMSTSNDR	JMS topic sender.
RTJSONSNDR	HTTP JSON sender.
SOAPSNDR	HTTP SOAP sender.

Indicate whether or not this sender is currently **Active**.

For JMS related senders, indicate whether the **MSG Encoding** is **ANSI message encoding** or **UTF-8 message encoding**. (Note that for all other types of senders, the Character Encoding context type is used to configure encoding.)

If the Message Class is one that connects to a JMS Queue or JMS Topic, indicate the appropriate **JMS Connection**

Fastpath: Refer to JMS Connection for more information.

If the Message Class is one that connects to a JMS queue, indicate the name of the **JMS Queue** to define where the message is to be sent.

Fastpath: Refer to JMS Queue for more information.

If the Message Class is one that connects to a JMS topic, indicate the name of the **JMS Topic** to define where the message is to be sent.

Fastpath: Refer to JMS Topic for more information.

Message Sender - Context

The sender may require context information to define additional information needed by the system to successfully send outgoing messages. Open **Admin** > **Integration** > **Message Sender** and navigate to the **Context** page.

Description of Page

Define the **Context Type** and **Context Value**, which contain parameters for senders when more information is required. See below for the supported context values for different types of senders.

Email Sender Context

The email sender is used by the business service that sends email messages real-time.

An email sender must point to the Message Class **RTHTTPSNDR**. In addition, the following context records should be defined for senders of this type.

Context Type	Description
SMTP Host name	The SMTP server host name.
SMTP Username	The user ID used to access the SMTP server.
SMTP Password	The password used to access the SMTP server.
Response Time Out	The amount of time the system should wait for a real time response.

Copyright © 2024, Oracle and/or its affiliates.

HTTP Sender

An HTTP sender is one that sends messages to an HTTP server using the HTTP protocol. HTTP senders should reference a Message Class of **RTHTTPSNDR**, **RTJSONSNDR** or **SOAPSNDR**.

Various parameters are required to establish a session with the target HTTP server. You specify these parameters by defining a collection of context values for the sender. A set of context types related to HTTP variables is provided with the product. The following section describes the context types and where appropriate, indicates valid values.

Before defining the HTTP sender, you need to find out how the HTTP server on the other side expects to receive the request, and in particular, to answer the following questions:

- What is the address of the HTTP server?
- Is the HTTP server using a POST or GET HTTP method?
- If the server is using POST, how are message contents passed? Does it use an HTTP FORM or does it pass the data in the body of an XML message?

Context Type	Description	Values
HTTP URL1 - URL9	Used to construct the URL of the target HTTP server. Since the URL may be long and complex, you can break it into smaller parts, each defined by a separate context record. The full URL is built by concatenating the values in URL1 through URL9. You may use substitution variables when entering values for URL parts. Note that the substitution string `@XMLMSG@` may be used for **GET** calls if an XSL has been applied to convert the message into HTTP GET parameters. It is useful if the HTTP Form is not applicable to the type of message. Refer to Support for Dynamic URLs for configuration needed to support dynamic URLs when sending an outbound message.	
HTTP Method	The HTTP method used to send the message. **Note:** The SOAP sender message class **SOAPSNDR** only supports the **POST** method.	**POST** or **GET**
HTTP Transport Method	Specifies the type of the message. You can either send the message or send and wait for a response.	**Send** or **sendReceive**
HTTP Form Data	Used when the message is in the format of an HTML Form (`Content-Type: application/x-www-form-urlencoded`). This context specifies the form parameters (data) that should be passed in the HTTP message. Since a form may have multiple parameters, you should add a context record for each form parameter. The value of a form parameter takes the format of *x=y* where *x* is the form parameter name and *y* is its value. If *y* contains the string @XMLMSG@ (case sensitive) then this string is replaced by the content of the service response XML message. The @XMLMSG@ string can be	

Copyright © 2024, Oracle and/or its affiliates.

Context Type	Description	Values
	used in the HTTP Form Data or in the HTTP URL, but not in both. If a context record of this type is defined for a sender, the sender uses the HTML Form message format to send the message even if @XMLMSG@ is not specified in one of the context records. If a context record of this type is not defined for a sender, then the XML is sent with `Content-Type: text/plain`. When using POST it is put in the HTTP message body. Always required when using the GET method. If you are using the GET method and do not specify a Form Data context record, no message is transferred to the HTTP server. You may use substitution variables when entering values for Form Data.	
HTTP Login User	The HTTP server may require authentication. Add a context record of this type to specify the login user to use.	
HTTP Login Password	The HTTP server may require authentication. Add a context record of this type to specify the login password to use.	
HTTP Header	Sometimes the HTTP server on the other side may require the addition of HTTP headers to the message. For each HTTP header that has to be specified you should add a context record with a value having the following format *x:y* where *x* is the header name and *y* is the value for the header	
HTTP Time Out	Indicates the amount of time to wait for a connection to be established with the remote system.	
Character Encoding	Indicates if the message should be encoded. The sender will add to the HTTP's content type header the string `;charset=x` where *x* is the value of this context and when sending the message it will encode the data in that encoding.	**UTF-8** or **UTF-16**
Response Time Out	The amount of time the system should wait for the remote system to send a response.	
Sender Security Type	Indicate the desired security type to apply. (Note that additional security types are applicable only to the SOAP Sender. See below for more information.)	**BASIC** (HTTP Basic), **TEXT** (Username Token plain text), **DIGEST** - Username Token Digest, **OWSM** - OAuth Security via OWSM, **OAUTH** - OAuth Security.
OWSM Policy Set	Applicable only if the Sender Security Type is **OWSM**. Defines the policy set to apply.	Enter a valid value for the extendable lookup Set of Policies (**F1-SetOfPolicies**). The product provides the value

Copyright © 2024, Oracle and/or its affiliates.

Context Type	Description	Values
		F1-OAUTH that may be used here.
OAuth Grant Type	Applicable only if the Sender Security Type is **OAUTH** . Defines the OAuth specification method for acquiring an access token.	**client_credentials**
OAuth Access Token URL	Used to construct the URL of the target authorization server. You may use substitution variables to construct the URL.	
OAuth Client ID	The generated identifier for the current application in the external system's authorization server.	
OAuth Client Secret	The secret paired with the OAuth Client ID as provided by the external system's authorization server.	
OAuth Key Ring	Defines the key ring used to store the OAuth Client ID and secret provided by the external system's authorization server. If a key ring is specified, the OAuth Client ID and OAuth Client Secret context entries are not allowed.	
OAuth REST API Scope	The authorization level granted by the end point.	
OAuth Token Request Details in Body	By default, the authorization token request includes the Client ID and Secret information in the message header. This context type may be used to indicate that this information should be sent in the body of the request instead.	Set this to 'Y' to send the Client ID and Secret information as part of the body of the authorization token request.

Real-time HTTP Sender

The following context type is only applicable to senders with the **RTHTTPSNDR** message class.

Context Type	Description
Content Type	Populate a value here to override the **Content-Type** attribute in the HTTP header, which defaults to **text/xml**.

SOAP Sender

A SOAP sender is an HTTP sender that automatically adds support for the SOAP format. For this type of sender (message class of **SOAPSNDR**), besides the context values listed above, the following context entries may also be supplied.

Context Type	Description	Values
Message Namespace URI	Used to indicate a specific namespace to be included in messages for this sender. Note that this value is used only when the External Message link to this sender is configured with a **Namespace Option** of **Configured on Sender**.	
SOAP Insert Timestamp	Indicate whether a timestamp should be added. Default value is 'N'.	**Y** or **N**

Copyright © 2024, Oracle and/or its affiliates.

Context Type	Description	Values
SOAP Expiration Delay (in seconds)	Indicate an expiration delay to add to the timestamp. The default value is 60.	
Sender Security Type	Additional security types are supported for SOAP senders for specific third party integrations.	**CIM** - Uses an appropriate namespace and adds security information in the SOAP header specific to Landis+Gyr, **MS_V30** - Uses an appropriate namespace and adds security information in the SOAP header specific to MultiSpeak v3.0, **MS_V41R** - Uses an appropriate namespace and adds security information in the SOAP header specific to MultiSpeak v4.1.**ITRON** - Uses appropriate SOAP header information security information in the SOAP header specific to Itron.

Note: Refer to Add SOAP Header Parameters at Runtime for information about dynamically including SOAP Header parameters when sending a message.

JMS Sender

A JMS sender is one that sends messages to a JMS queue or JMS topic. JMS senders should reference a Message Class of **RTJMSQSNDR** or **RTJMSTSNDR**, respectively.

The following parameters are used to connect to the JMS resource.

Context Type	Description	Values
JMS Message Type (Bytes(Y)/ Text(N)	Indicates whether the data is sent as a bytes message or as a text message.	**Y** or **N**
JMS User Name	Enter the user name to connect to the JMS resource.	
JMS User Password	Enter the password to use to connect to the JMS resource.	
JMS Header	If JMS header values are required for the message, use this context type. For each JMS header that has to be specified, add a context record with a value having the following format *x:y* where *x* is the header name and *y* is the value.	

Application Variables

Some APIs may require application identifying variables to be included in the message payload. These values are typically available at provisioning time and need to be captured in relation to these API configurations.

Note: This type of context is not involved in the process of orchestrating and routing the API call. It needs to be incorporated into the payload by the application logic that composes the message using the "Get Application Variables (F1MsgVars)" service script.

The following parameters may be used to capture application variables.

Copyright © 2024, Oracle and/or its affiliates.

Context Type	Description	Values
Application ID	Identifies the current environment for this API.	Determined at provisioning time by the API provider.
Tenant ID	Identifies the current tenant for this API.	Determined at provisioning time by the API provider.

Defining Outbound Message Types

Refer to Outbound Messages for an overview of this functionality.

Use this page to define basic information about an outbound message type. Open this page using **Admin** > **Integration** > **Outbound Message Type.** You are brought to a query portal with options for searching for outbound message types.

Note: This page is not available if the **External Message** module is turned off.

Once an outbound message type has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The **Outbound Message Type** zone provides basic information about an outbound message type. Besides the code and description, the outbound message type captures the outbound message business object and the relative priority of the outbound message. The priority is only applicable if the outbound message is processed using OSB. In addition, if the outbound message type is linked to any web service categories, they are displayed.

CAUTION: Important! When adding new records, carefully consider the naming convention of the outbound message type code. Refer to System Data Naming Convention for more information.

Note that in addition to standard actions available on this portal, there is also a special **Add to Category** button in the page action area. Click this button to link an outbound message type to one or more web service categories.

The **External System Links** zone is visible if the outbound message type is linked to any external systems.

External Systems

Use this page to define an external system and define the configuration for communication between your system and the external system.

External System - Main

Open this page using **Admin** > **Integration** > **External System.**

Note: This page is not available if both the **External Message** and the **Open Market Interchange** modules are turned off.

Description of Page

Enter a unique **External System** and **Description**.

CAUTION: Important! When adding new records, carefully consider the naming convention of the external system code. Refer to System Data Naming Convention for more information.

Use the field **Our Name In Their System** to specify the identity of your organization (i.e., your external system identifier) in the external system.

Note: The workflow process profile and notification download profile are only applicable to products that support workflow and notification. They and are not visible in the product if the **Open Market Interchange** module is turned off.

If this external system sends inbound communications through notification upload staging, the type of workflow process that is created is controlled by the sender's **W/F (Workflow) Process Profile**.

If you send notifications to this external system, select a **Notification DL** (download) **Profile** that is used to define the configuration of the outgoing messages.

Copyright © 2024, Oracle and/or its affiliates.

> **Note:** The remaining fields are not visible if the **External Message** module is turned off.

Set **Usage** to **Template External System** for external systems whose outbound message type configuration is inherited by other external systems.

If the outbound message type configuration should be inherited from a template external system, define the **Template External System**. If this field is specified, the outbound message type collection displays the data defined for the template system as display-only.

The **Outbound Message Type** accordion contains an entry for every type of outbound message defined for this external system. For each type of outbound message identify its **Outbound Message Type**.

Define the **Processing Method** for messages of this type. Valid values are **Batch**, **Real-time**, **SOA** and **XAI**.

Define an appropriate **Message Sender** if the processing method is **XAI** or **Real-time**.

Namespace Option is used when your message should include a namespace in the resulting XML. The valid option are **Standard Namespace** and **Configured on Sender**. If the value is Standard Namespace, the system will generate a namespace for the resulting WSDL with the following value: **http://ouaf.oracle.com/outbound/AAA_BBB**, where AAA is the external system code and BBB is the outbound message type code. If the value is Configured on Sender, then the value of the namespace is taken from the sender context with context type **Message Namespace URI**.

Define an appropriate **Batch Control** if the processing method is **Batch**.

If the message sender is one with a message class of **RTJSONSNDR**, indicate the **JSON Conversion Method**. The valid values are **Base JSON Conversion**, **Standard API Conversion** and **XSL Transformation**. Refer to Real Time Message Configuration for more information about these methods and the additional configuration that may be applicable.

If the **JSON Conversion Method** is **Base JSON Conversion**, the **Request Schema** is enabled. Populate a data area that defines the schema for the XML format to convert the outbound message's BO schema to prior to performing the JSON conversion. Refer to Real Time Message Configuration for more information.

The **Message XSL** is the schema used to transform information from the format produced by the system to a format understood by the sender, who receives a message of this type. This is not applicable for Processing Method of **SOA**. Refer to Referencing an XSL for information about configuring where the XSL is defined.

Enter the file name of the appropriate **W3C Schema** if you want to validate the message built for outbound messages for this external system / outbound message type prior to being routed to their destination. Refer to Outbound Message Schema Validation for more information. This is not applicable for Processing Method of **SOA**. This feature is not available for cloud implementations.

If the **JSON Conversion Method** is **Base JSON Conversion**, the **Response Schema** is enabled. Populate a data area that defines the schema for the XML format that the JSON message is initially converted to. The XML is then converted to BO XML. Refer to Real Time Message Configuration for more information.

Response XSL will only be displayed when the processing method is **Real-time**. Refer to Real Time Messages for more information on how it is used. Refer to Referencing an XSL for information about configuring where the XSL is defined.

External System - Template Use

If you are viewing an external system whose usage is a **Template External System**, use this page to view the other external systems that reference this one. Open this page using **Admin** > **Integration** > **External System** and then navigate to the **Template Use** tab.

Description of Page

The tree shows every external system that references this external system as its template.

Message Option

The Message Option page defines various system settings used by the system when processing external messages.

To define options for your environment, open **Admin** > **Integration** > **Message Option**.

Description of Page

Copyright © 2024, Oracle and/or its affiliates.

The following options are supported.

Option	Description	Option Name
Default Email Sender	This is the default Message Sender used for sending e-mails when no explicit Message Sender is specified.	**defaultEmailSender**
Default User	The default user is used by the system to access your product when no other user is explicitly specified.	**defaultUser**
Email Attachment File Location	This is the default location of e-mail attachment files. If not specified, the e-mail service provided with the product assumes a full path is provided with each attachment file.	**emailAttachmentFileLocation**
Email XSL File Location	This is the default location of e-mail XSL files. If not specified, the e-mail service provided with the product assumes a full path is provided to an XSL file as part of an e-mail request.	**emailXSLFileLocation**
Messages Language	The default language to use for the messages.	**language**
Outbound Message Schema Location	Enter the full path of the virtual directory where valid W3C schemas are stored if your implementation wants to validate outbound message schemas. For example: http:/localhost/cisxai/schemas.	**xaiOuboundSchemaLoc**
Schema Validation Flag	Enter **Y** to turn on schema validation for outbound messages. Enter **N** to turn this off.	**xaiSchemaValidationCheck**

Note that the following options are no longer applicable.

Option
To Do Type for Inbound JMS Message Errors
To Do Type for Outbound Message Errors
XSD Compliance

Managing Outbound Messages

Use this page to view information about outbound messages.

Outbound Message - Main

Open this page using **Menu** > **Integration** > **Outbound Message**.

Description of Page

Outbound Message ID is the system-assigned unique identifier of the outbound message. These values only appear after the outbound message is added to the database.

Copyright © 2024, Oracle and/or its affiliates.

The **Processing Method** indicates whether this record will be processed by a **Batch** extract process or **Real-time**. (The value of **XAI** is no longer supported). The value defined on the external system / outbound message type collection populates this value.

When records are created with a processing method of **Batch**, the system sets Extract to **Can Be Extracted**. Change the value to **Not to be extracted** if there is some reason that processing should be held for this record.

For records with a processing method of **Batch**, **Batch Control** indicates the process that will extract this record. This value is populated based on the on the external system / outbound message type's value. **Batch Number** indicates in which batch run this record was extracted or will be extracted.

The **Retry Count** is no longer applicable..

The **Creation Date** indicates the date that this record was created.

The status is no longer applicable.

Outbound Message - Message

Use this page to view the XML source used to build an outbound message. Open this page using **Menu** > **Integration** > **Outbound Message** and then navigate to the **Message** tab.

Description of Page

The **XML Source** is displayed.

If a message XSL is defined on the external system / outbound message type record linked to this outbound message, the **Show XML** button is enabled. Click this button to view the XML that is a result of applying the Message XSL to the XML source.

Outbound Message - Response

Use this page to display the XML response. Open this page using **Menu** > **Integration** > **Outbound Message** and then navigate to the **Response** tab.

Description of Page

The **XML Response** and optionally the **XML Raw Response** is displayed.

XML Response displays the response data from the system called by the real-time message. If a response XSL is defined on the external system / outbound message type record linked to this outbound message, a transform is performed and the XML Raw Response displays the original, unchanged response.

Web Service Adapters

The base product provides a configuration object called Web Service Adapter that is used to help build configuration objects to allow for functionality in the system to initiate a web service call from within the system. A Web Service Adapter provides the following functionality:

- WSDL (web service description language) import. An implementer can use the WSDL import functionality to read the details of a WSDL into the system
- Internal API generation. The system generates internal data areas that have two main purposes: they provide the API for custom code to define the appropriate input and they provide output data for the web service call using Oracle Utilities Application Framework schema language. In addition, the web service dispatcher uses element mapping defined in the data areas to transform the internal XML into the structure expected by the external system as described in the WSDL.
- Defines the URL needed to perform the web service call at runtime.

Understanding Web Service Adapters

The following topics describe the system functionality in more detail.

Copyright © 2024, Oracle and/or its affiliates.

Importing a WSDL

Configuring a Web Service Adapter starts by identifying the WSDL (the web service description language document used to define the interface) that will be provided by the external system. The following steps describe the base product functionality provided to allow a user to import a WSDL.

- Navigate to the **Web Service Adapter** page in add mode and select the appropriate base business object.
- Enter a meaningful Web Service Name and appropriate descriptions.
- Provide the URL of the given WSDL.
- Click **Import** to retrieve the details of the WSDL. The system then parses the WSDL details and populates the WSDL Service Name, WSDL Source, WSDL Port, URL and a list of Operations (methods) defined in the WSDL.
- Determine which Operations should be **active** based on the business requirements for invoking this web service. **Active** operations are those that the implementation is planning to invoke from the system. These require appropriate request and response data areas generated for them. The following section provides more information about that.
- Specify the appropriate Security Type to configure the type of security to use when invoking this web service.
- Click Save.

At this point, a web service adapter record is created in pending status. The next step is to generate the request and response data areas for the operations configured as active.

Generating Request and Response Data Areas

Each **active** operation for the web service adapter requires a pair of data areas, request and response, that represent the request and response XML messages for the operation.

The base product provides steps to generate the data areas as follows:

- As described in the Importing a WSDL section above, the operations listed in the WSDL are generated for the web service adapter and the implementer should indicate which operation to activate.
- After saving the **pending** web service adapter, the display lists all the active operations and for each one includes a **Generate** button.
- After clicking **Generate** for an operation, a window appears where the names of the new Request and Response Data Areas may be defined. Click **Save** to generate the data areas.

The generated data areas provide the API for the implementer to use when implementing the web service call in an appropriate algorithm or service in the system. The data areas contain the appropriate mapping from the elements that the implementer works within the code that invokes the web services and the WSDL definitions.

To facilitate generating the request and response data areas, the base product invokes a special business service used to create the appropriate mapping. The business service is defined as a BO option on the Web Service Adapter business object. This allows an implementation to provide a custom business service to further enhance the request and response mapping where appropriate.

Note:

Generated data areas. It is possible to edit and modify the generated data areas after they are created. An implementer can change element names or remove unneeded elements if desired. Manually changing the generated data areas must be done only when absolutely necessary. This is because the system is not able to validate manual changes and issues with the data areas would only be detected at run time.

Activating Web Service Adapters

The business objects provided by the base package for web service adapters include a simple lifecycle of **Pending** and **Active**. Configure the web service adapter and its data areas while in **Pending** status and activate it when it is ready to be implemented in the appropriate system functionality.

Copyright © 2024, Oracle and/or its affiliates.

Invoking Web Services

To make a call to a web service using a web service adapter, the system has provided a Web Service Dispatcher business service (**F1-InvokeWebService**) to submit a web service call. The calling program is responsible for retrieving all the information to correctly populate the request data required by the web service call before invoking the business service.

> **Note:**
>
> Refer to the detailed description of the business service for more information.

Limitations

The following points highlight limitations associated with the types of web services that the system supports:

- It is possible for one WSDL document to contain definitions for several web services. The system currently supports only one port or service per WSDL document.
- It is possible for a WSDL to support multiple message patterns. The system currently supports only request / response.

Setting Up Web Service Adapters

Use the Web Service Adapter portal to define the configuration needed to communicate with an external system using a web service call. Open this page using **Admin** > **Integration** > **Web Service Adapter.** You are brought to a query portal with options for searching for web service adapters.

Once a web service adapter has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The **Web Service Adapter** zone provides basic information about the web service annotation type.

Please see the zone's help text for information about this zone's fields.

> **Fastpath:** Refer to Understanding Web Service Adapters for information about common web service adapter functionality.

Sending Email

The framework provides the ability to initiate an email from within the system. The following topics highlight the functionality available.

- Sending email "real time" using a specific business service. The framework provides a business service **F1-EmailService** that supports sending an email. The schema supports elements for all the information required to create an email real time. The SMTP information (host, user name and password) may be provided or may be defined on a message sender, that may be provided as input. In addition, the business service supports using a default message sender defined as a message option. Review the business service schema for information about the input elements.

  > **Note: Retry setting**. An option in the system properties file allows your implementation to configure the number of times to retry (if any) if the SMTP server is unavailable. Refer to the server administration guide for more information.

- The business service supports sending attachments. The may either reference attachments from the file system (file name and content ID) or reference attachment records captured in the system's attachment table. Note that attachments in the file system are not supported in a cloud implementation.

  > **Note:** Validating attachments. If a Validate Email Attachment algorithm is plugged into the installation record, it is called to validate the attachments supplied, if applicable.

- Using an outbound message to send an email. This option allows for different variations as described in Outbound Messages.

Copyright © 2024, Oracle and/or its affiliates.

- Some emails may be created en masse (for example a large group of emails routed to users for a given set of To Do entries). In this case, the records can be created in the staging table for processing using OSB.
- Messages may still be sent real time using one of two business services described in Real Time Messages. This option is an alternative to the dedicated email service described above when aspects of the outbound message functionality are needed, such as the ability to instantiate a record as an audit or to include additional logic via BO plug-ins as part of sending the email.

Web Service Category

The product provides the ability to categorize web services that are defined in the system. The term web services refers to Outbound Message Types, Inbound Web Services, and for those implementations where they are applicable, XAI Inbound Services.

A given web service may be linked to more than one category. The product supplies web service categories and most base delivered web services are configured with appropriate categories. Implementations may define new categories and may link custom web services to base delivered categories or to custom categories. In addition, implementations may link additional categories to base delivered web services. However, implementations may not remove base delivered web services from base delivered categories.

If your implementation uses the Integration Cloud Service Catalog, the integration provides the categories for each web service that is sent to the catalog.

The web service category portal supports adding and removing web services from the category. In addition, the Outbound Message Type portal and Inbound Web Service portal each provide a page action button to **Add to Category**.

Defining Web Service Categories

Refer to Web Service Category for an overview of this functionality.

Use this page to define basic information about a web service category. Open this page using **Admin** > **Integration** > **Web Service Category.** You are brought to a query portal with options for searching for web service categories.

Note: This page is not available if the **External Message** module is turned off.

Once a web service category has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The **Web Service Category** zone provides basic information about a web service category.

CAUTION: Important! When adding new records, carefully consider the naming convention of the web service category code. Refer to System Data Naming Convention for more information.

Once a web service category exists, an **Add Web Services** zone is provided to find and select web services to link to the web service category.

When viewing a category that already has web services linked to it, they are visible in the zone **Included Web Services** . This zone allows a user to remove a customer owned web service from the list.

JMS Message Browser

The JMS Message Browser portal allows you to select a JMS queue and view messages currently in the queue.

In order for a JMS queue to be available on the portal, a message sender must be defined that is configured for the appropriate JMS queue with the credentials to connect to the queue.

Copyright © 2024, Oracle and/or its affiliates.

- If your organization sends real-time outgoing messages to a JMS queue, this configuration would exist as per the details in Real-Time Message Configuration.
- If inbound web service messages are routed to the system via a JMS queue, no configuration is needed in the system. However, if you would like to view the messages in the queue in the JMS message browser portal, configuration for the JMS queues as described for outgoing messages is required.

Navigate to the portal using **Main** > **Integration** > **JMS Message Browser.**

The JMS Senders zone provides a list of configured message senders eligible for selection.

The JMS Message List zone is visible for the JMS Sender broadcast from the first zone. This zone supports selecting one or more records to delete from the queue. Use the message selector to limit the results to messages that satisfy the message selector. This uses standard JMS API message selector functionality. Refer to the zone embedded help for information about the supported syntax.

The JMS Message Details zone displays a message broadcast from the list zone.

Oracle Integration Cloud Catalog

Oracle Integration Cloud (OIC) is an offering that serves as integration infrastructure for Oracle cloud solutions. The product provides an adapter for OIC to streamline integration between your edge application and OIC.

The product provides a mechanism (referred to as an adapter) for OIC to retrieve the REST catalog and another to retrieve the SOAP catalog from a given product. The expectation is that a given implementation would use either REST or SOAP and not both, but for those implementations that do support both classes of web services, OIC can retrieve both catalogs.

For the SOAP catalog, the adapter retrieves the name, the source system, the WSDL location and namespace for each web service.

For the REST catalog, the adapter retrieves the name, the source system and the OpenAPI specification for each web service.

It is possible that not every web service supported by an edge product is managed by OIC. In order to only include the appropriate web services in the adapter, configuration is needed to identify which web services to include.

The web service catalog is used to identify the records that should be retrieved by the adapter. Separate catalogs exist for REST and for SOAP

- For SOAP inbound messages, the system supports both the use of inbound web services and XAI inbound services that are deployed via IWS. Each IWS or XAI inbound service that should be included in the catalog must be flagged in the SOAP catalog. Note that only deployed services are returned to the catalogue.
- For REST inbound messages, each IWS that should be included in the catalog must be flagged in the REST catalog.
- For outbound message, the system requires the creation of an External System that includes each outbound message type that the external system receives. For outbound messages that are integrated through OIC, the external system itself will represent OIC. Rather than identifying each outbound message type to include in the catalog, only the external system needs to be flagged. The adapter will return web service information for all the outbound message types configured for the external system. If an implementation supports both REST and SOAP, separate external systems should be created to separately group the REST and SOAP outbound messages.

Note: The subordinate catalog functionality is no longer applicable and will be removed in a future release.

Fastpath: Refer to Maintaining the Web Service Catalog for more information.

Web Service Catalog Configuration

The topics in this section describe the configuration needed in your edge application to integrate with OIC.

Copyright © 2024, Oracle and/or its affiliates.

Web Service Catalog Master Configuration

The **Service Catalog Configuration** (master configuration) record defines several system wide settings related to integrating with the service catalog.

It includes configuration for the WSDL base URL for the SOAP catalog. Note that this element supports the functionality described in Referencing URIs. Note that for the REST catalog, the system will use the URL defined in the substitution variable **F1_OPEN_API_BASE_URL**

The subordinate server section is no longer applicable and will be removed in a future release.

For more information about specific fields in the master configuration, refer to the embedded help.

Maintaining the Web Service Catalog

Refer to Oracle Integration Cloud Catalog for an overview of web service catalog functionality.

To add or remove services that are reported to the OIC catalog, navigate to the portal using **Admin** > **Integration** > **Web Service Catalog.**

The Catalog by Class zone provides the list of classes (REST and SOAP). Broadcast the web service class that you want to work with.

The Catalog zone provides a list of services that are currently in the catalog based on the web service class chosen. Users may use this zone to remove services from the catalog.

The Candidate Services zone provides a list of external systems and inbound web services that are not currently linked to the catalog. Users may use this zone to add services to the catalog. The inbound web services will be limited to those with the same web service class as the one broadcast. All external systems are shown regardless of the web service class because there is no web service class configuration on external systems.. The expectation is that implementers will create the external system to include either only REST or only SOAP services and will know which external system to include in which catalog. For the SOAP web service class, if XAI inbound services are applicable, they are also visible.

Note: For inbound SOAP web services and XAI inbound services, they may be selected for the SOAP catalog at any time. However, only deployed services are returned to the catalog by the OIC adapter.

XAI Documentation Note

The XAI functionality is legacy functionality and not recommended for new implementations. The help topics are no longer provided.

Chapter 11

Integrations

Topics:

- Creating Application Links in External Applications
- LDAP Integration
- Oracle Identity Manager Integration
- Provision Users with SCIM Open Standard
- User Redirection
- Batch Scheduler Integration
- Data Synchronization
- Operational Analytics
- Analytics Integration
- Business Flags

This chapter provides high level information about product integrations supported for all products that use Oracle Utilities Application Framework.

Creating Application Links in External Applications

When integrating Oracle Utilities applications with other external applications, you can create links from external applications to specific portals and pages within the Oracle Utilities applications (also known as "deep linking"). These links typically leverage Business Process Assistant (BPA) scripts to help navigate the user to the appropriate target portal or page and to make provided parameter values available for use on the portal or page.

> **Note:** This technique may also be used within the product to launch a new browser tab from within the application when it's important to maintain the current screen. For example, launching the GIS Viewer in Oracle Utilities Work and Asset Management by creating a new browser tab.

To do this, parameters are added to the application URL by appending a question mark (?) to the end and using the syntax "key=value". If you require more than one parameter, use an ampersand (&) to separate each key=value pair. There are several predefined parameter key values recognized by the application. In addition, specific parameters understood by the script may be provided. The following list highlights the predefined parameters.

- **location=**<navigation key> - If the exact page to launch is known, the navigation key value of that page can be part of the URL using 'location='. For example, if an approval process has sent an email to a user to approve an adjustment, the link configured in the email may include the adjustment's navigation key.
- **script=**<BPA script name> - In almost every use case, a script is needed to help navigate to the correct page and display the correct information. Even if a location is known and provided in the URL, if there is context data provided as a parameter, it is the responsibility of a script to move that parameter data from the URL into the page data model, so that the page can access the information and display the correct record. In cases where the target page is dynamic based on information provided by parameters, the script may be used to determine what page to launch.
- **portal=true** - When using the location parameter where the target page is a portal page and passing context data, use this parameter. The allows the system to know that the parameter data must be pushed into portal context.
- **initNav=false** - By default, if a script is provided but the location attribute is not provided, the system navigates to the user's home page prior to executing the script. If the script itself includes a step to navigate to a target page as one of its initial steps, the navigation to the home page is unnecessary and may degrade performance. This optional attribute may be included in the URL to bypass the home page.

  > **Note:** The system still requires a page to be launched for technical reasons. A blank portal with no zones is used for this purpose. Users may see this portal (called **Launching Application**) briefly before the navigation initiated by the script. In addition, this is the portal that the user will remain on if there are any errors in the script or if the script does not navigate anywhere.

- **minSidebar=true** - This option minimizes the sidebar area as part of launching the application. (The user will be able to maximize the sidebar themselves if needed). You would use this option to potentially improve the speed at which the application is launched. Whether this is useful will depend on the use case. If the reason for launching the application is to bring the user to a specific record to view details about that record, the sidebar may be providing valuable information about the record. In this case, it may be correct to display the sidebar by default (and not include this parameter in the URL). On the other hand if the use case is that a user is already in the application and clicks a link that opens a new browser tab to show the user specific detail about a chosen record, while keeping the original page open on the first tab, the sidebar may contain the same information already visible on the original tab. In this case, minimizing the sidebar may be beneficial.

  > **Note:** For backward compatibility, the system still supports the original parameter **minDashboard=true**, which has analogous functionality.

For example, the following URL illustrates launching the script **CM-StopAutoPay** for a given account, assuming the application URL for launching the system is http://system-server:1234/cis.jsp:

- **http://system-server:1234/cis.jsp?script=CM-StopAutoPay&ACCT_ID=1234512345&location=accountMaint**

It doesn't matter in which order the parameters are provided. The system processes them in the correct order. For example, the following examples are processed by the system in the same way:

Copyright © 2024, Oracle and/or its affiliates.

- **http://system-server:1234/cis.jsp?ACCT_ID=1234512345&script=CM-StopAutoPay&location=accountMaint**
- **http://system-server:1234/cis.jsp?ACCT_ID=1234512345&location=accountMaint&script=CM-StopAutoPay**

These parameters are kept in a common area accessible by any script for the duration of the session. To use these parameters on a script you may reference the corresponding **%PARM-<name>** global variables. In this example, after the system is launched any script may have access to the above account ID parameter value by means of the **%PARM-ACCT_ID** global variable. Also note, these parameters are also loaded into temporary storage (to continue the example, there would also be a temporary storage variable called **ACCT_ID** that holds the passed value).

Determining the Application URL

When building the link from within application logic, the service script **F1-EnviURL** may be used the retrieve and build the application URL information.

Navigate to a Given Record's Maintenance Portal

The product provides a script **F1-GotoPrtl** (Navigate to portal for an MO and key values) that may be used for any use case that simply navigates to the maintenance page for a given record and displays that record. This script is only applicable to records that are governed by a business object (and define a navigation option as a BO option). It requires the parameters **mo=** with the maintenance object code of the record and the parameters **pkValue1=** through **pkValue5=** for providing the key values. It looks up the appropriate portal navigation option from the record's BO, populates keys into the 'page data model' and navigates to the portal.

It is also recommended to include the **initNav=false** attribute when building a URL with this script. The example below navigates to the appropriate Migration Data Set portal for the given ID's business object.

- **http://system-server:1234/cis.jsp?script=F1-GotoPrtl&pkValue1=1234512345&mo=F1-MIGRDS&initNav=false**

LDAP Integration

Organizations commonly use a Lightweight Directory Access Protocol (LDAP) security repository as a source of security credentials. The system provides support for importing users and groups from an external LDAP repository into the product to populate Users and User Groups in the system. Once imported, all user and group functions are available. You can resynchronize your LDAP users and groups at any time.

Note: Import only. The system currently supports importing LDAP information. Exporting users and groups from the system to LDAP is not provided.

Note: Additional configuration. When importing new users and / or groups, additional configuration is needed in the base product. For example, after importing a new user group and its users, the user group configuration should be updated to define the valid application services for the user group. After importing a new user, additional configuration may be needed on the user such as valid To Do Roles, valid Home Page, etc.

Fastpath: Refer to Users and User Groups for more information about the application security and what it controls.

This section the functionality provided in the framework application that supports LDAP. Refer to the *LDAP Integration* technical reference paper for more information about typical steps related to the full integration.

LDAP Integration Overview

This topic provides a high level overview of the integration process.

Copyright © 2024, Oracle and/or its affiliates.

At a high level, the base product provides a process to import user group and / or user definitions from and LDAP repository. This is a one way integration.

- When importing a user, if it is not already found in the system, it will be added; otherwise its attributes will be updated according to the imported information.
- When importing a user group, if it is not already found in the system, it will be added; otherwise its attributes will be updated according to the imported information.
- When importing a user, its user group links will be updated as per the information in the import file. In addition, if there are any user groups linked to the user that are not found in the system, they will be added (however, the other users linked to that group in the LDAP repository will not be added as part of this step).
- When importing a user group, its user links will be updated as per the information in the import file. In addition, if there are any users linked to the user group that are not found system, they will be added (however, the other user groups linked to that user in the LDAP repository will not be added as part of this step).
- The import will not cause any deletions of the User or User Group to occur.

A Batch Process Initiates the Import

A batch process is used to initiate the import of information from the LDAP repository. **F1–LDAP** may be submitted ad hoc or may be set up in a scheduler to periodically re-sync the information from the LDAP repository into the application.

The batch process uses parameters to define how to connect to the LDAP repository. In addition, parameters are used to indicate which user or group is being imported.

Adjusting Data to Import

The system provides several mechanisms for adjusting data that is being added to the system:

- There is an **LDAP Import Preprocess** algorithm plug-in spot on the installation record. Algorithms plugged in here are called by the batch process prior to the add or update of any records. It may be used to make adjustments to the data before doing updates in the application.
- Specifically for creating or updating Users, the **F1–IDMUser** business object is used to add and create the user. The standard BO Preprocessing algorithm plug-in spot may be used to adjust data prior to creation. Note that the system does not support defining a characteristic to support a template user that can aid in populating configuration for a new user. However, this integration can take advantage of using a default template user as an algorithm parameter. Refer to Template User for more information.
- The LDAP mapping file supports some attributes to perform simple modifications to data.
  - The **transform** attribute supports values to truncate values or to convert data to upper case.
  - The **autoGenerate** attribute is specific to the User ID field. Setting this to true will trigger code that will automatically populate the User ID based on the user's name. Refer to LDAP Mapping for more information.

Performing Additional Processing After Import

The system provides a plug-in spot on the installation record called **LDAP Import**. Algorithms plugged into this spot are called after users or user groups have been added or updated. It may be used to perform any extra processing that may need to be executed.

Configuring LDAP Integration

To interface the LDAP based security repository with the authorization component of the Oracle Utilities Application Framework product the following must be performed:

- The location and port number of the LDAP based security repository must be defined to in the JNDI Server.
- • The LDAP based security repository must be mapped to the Oracle Utilities Application Framework security model. This mapping is expressed as an XML file containing the LDAP query, rules and defaults used in the transformation.
- • The mapping file must be configured on the **F1-LDAP** batch job.

Copyright © 2024, Oracle and/or its affiliates.

Define the JNDI Server

The first step in the configuration process is to define the location of the LDAP based security repository server so that the interface can connect to the physical attributes of the interface. This is done by creating a JNDI Server.

Note: The LDAP server is strictly not a JDNI source but is treated as a JNDI source for the integration.

Enter a reasonable JNDI Server name and description.

Populate the **Provider URL** using the format **ldap://<hostname>:<portnumber>** where **<hostname>** is the host of the LDAP server and **<portnumber>** is the port used for the interface.

For the **Initial Context Factory**, the interface uses the standard **com.sun.jndi.ldap.LdapCtxFactory** provided with java for the LDAP interface. If your vendor supplies a custom context factory it may be used. Refer to the documentation provided with your LDAP based security repository for further information.

Define Mapping

The critical component of the interface is a file that describes the mapping between the LDAP based security repository and the system's security model. This file contains the mapping, rules and queries used by the LDAP batch program to provide the interface. The LDAP batch job includes the reference to the mapping file as a parameter. Refer to LDAP Mapping for more information on defining the mapping file.

Configure LDAP Batch Process

At this point, many parameters for the **F1–LDAP** batch control can be updated with system wide configuration.

- JDNI Server, User and Password may all be configured appropriately. Note that it is recommended that the **Security** setting for the Password be set to **Encrypt**.
- The **LDAP Configuration File** should be populated with the name and location of the LDAP Mapping file.
- If the LDAP service has any limitation to the number of objects that may be imported, configure the **LDAP Query Page Size** parameter to enable querying.

Note: Group and User Parameters. The assumption is that the Group or User input parameters are specific to a given import request and as such would not be populated as part of a configuration step.

Note: L2 Cache. The LDAP Import batch process requires the L2 Cache to be disabled since it needs to perform some updates in the outside of the worker threads. Any environment using LDAP Import must set **spl.runtime.batch.L2CacheMode=OFF** in the **threadpoolworker.properties** file. It is recommended to run the LDAP import in its own dedicated threadpoolworker.

LDAP Mapping

An LDAP repository consists of multiple entries. Each entry represents an object in the directory that is identified by a Distinguished Name (DN) and may contain one or more attributes. In a typical LDAP repository there is usually an entry for users and an entry for groups. The connection between users and groups may be implemented in two different ways:

- The users belonging to a group are defined in a special multiple-value attribute on the Group entry.
- The groups to which a user belongs are defined in a special multiple-value attribute on the User entry.

The mapping between LDAP security objects and base security objects is stored in an XML document that can be processed by the LDAP import batch job. As part of setting up your system for LDAP import, you need to define this mapping. The base package provides a sample mapping file called **ldapdef.xml** that can be used as a starting point and changed per your business requirements and your particular LDAP repository.

Once you have defined the mapping XML document, this is configured as a parameter in the **F1-LDAP** batch job.

The XML structure:

- The **LDAPEntry** element maps the LDAP entries to system objects (User or Group). The mapping file must contain one and only one LDAPEntry element for User and one for Group.

Copyright © 2024, Oracle and/or its affiliates.

- The **LDAPCDXAttrMapping** element within the LDAPEntry element maps attributes in the LDAP entry to attributes in the system object.
- The **LDAPEntryLinks** element describes objects linked to the LDAP entry. When mapping the user entity you need to describe how the groups the user belongs to are retrieved. When mapping the group entity you need to describe how the users contained in the group are retrieved.

The following table describes the attributes to define for each element.

Element	**Attribute**	**Description**
LDAPEntry	name	The name of the LDAP entry: - **Group** - **User**
	baseDN	The base distinguished name in LDAP fo
	cdxEntity	The name of the base product entity to w - **Group** - **User**
	searchFilter	An LDAP search filter that is used to loca replaced by the value from the user or gr
	Scope	Sets the scope of the search. Valid values - onelevel (the value normally used) - subtree
LDAPCDXAttrMapping	ldapAttr	The name of the LDAP attribute to be m to allow one LDAP element to map to n address should be used both for the Logi
	cdxName	The name of the base product attribute to For User, this is the element within the F For Group, this is either the 'group' or th
	default	The default value that will be assigned t one of the following occurs: - The LDAP attribute contains a null or e - The LDAP attribute does not exist or is Default values are applied only when cre
	autoGenerate	Set this to **true** in order to turn on auto define the user id as <first initial of first digits. If an existing user is found for the appended to the end). The system will in
	transform	Use this attribute to indicate if a transfor **truncate**. Note that this attribute should n

Copyright © 2024, Oracle and/or its affiliates.

Element	Attribute	Description
LDAPEntryLink	linkedToLDAPEntity	The name of the linked entity (**User** or **C** Group when describing the User entity.
	linkingLDAPAttr	The multiple-value attribute name on the
	linkingSearchFilter	The search filter to be applied to retrieve (&(objectClass=group)(memberOf= The search filter may contain the string % run time by the value of the attribute nam describing is a Group and the string is %r of the group you are importing. If the LI %dn%, it is replaced by the "dn" attribute
	linkingSearchScope	Sets the scope of the search. Valid values - **onelevel** (the value normally used) - **subtree**

Sample Mapping

The following XML describes a sample mapping. The example makes the following assumptions:

- The base product attribute **displayProfileCode**is defaulted to "NORTHAM" when adding a new user.
- The LDAP Group entry contains the list of users belonging to the group in the **departmentNumber** attribute.
- The groups to which a user belongs are retrieved by applying a search filter.

```
<LDAPEntries>
  <LDAPEntry name=" User" baseDN="ou=people,dc=example,dc=com" cdxEntity=" -
   user" searchFilter=" (&amp;(objectClass=inetOrgPerson)(uid=%searchParm%))">
    <LDAPCDXAttrMappings>
      <LDAPCDXAttrMapping ldapAttr="uid" cdxName=" user" -/>
      <LDAPCDXAttrMapping ldapAttr="cn" cdxName="externalUserId" -/>
      <LDAPCDXAttrMapping cdxName="language" default=" ENG" -/>
      <LDAPCDXAttrMapping ldapAttr="givenName" cdxName="firstName"/>
      <LDAPCDXAttrMapping ldapAttr="sn" cdxName= -"lastName"/>
      <LDAPCDXAttrMapping cdxName="displayProfileCode" default="NORTHAM" -/>
      <LDAPCDXAttrMapping cdxName="toDoEntriesAge1" default="30" -/>
      <LDAPCDXAttrMapping cdxName="toDoEntriesAge2" default="90" -/>
      <LDAPCDXAttrMapping cdxName="userEnable" default="ENBL"/>
    </LDAPCDXAttrMappings>
    <LDAPEntryLinks>
      <LDAPEntryLink linkedToLDAPEntity="Group" linkingLDAPAttr="departmentNumber" -/
>
    </LDAPEntryLinks>
 </LDAPEntry>
  <LDAPEntry name="Group" baseDN="ou=people,dc=example,dc=com" cdxEntity=" -
   Group" searchFilter=" (&amp;(objectClass=organizationalUnit)(ou=%searchParm%))">
    <LDAPCDXAttrMappings>
      <LDAPCDXAttrMapping ldapAttr="name" cdxName="Group" -/>
      <LDAPCDXAttrMapping ldapAttr="description" cdxName=" Description" -
default="Unknown" -/>
    </LDAPCDXAttrMappings>
    <LDAPEntryLinks>
      <LDAPEntryLink linkedToLDAPEntity="User" linkingSearchFilter=" -
      (&amp;(objectClass=inetOrgPerson)(departmentNumber=%distinguishedName%))" -
     linkingSearchScope="onelevel" -/>
    </LDAPEntryLinks>
  </LDAPEntry>
```

Copyright © 2024, Oracle and/or its affiliates.

```
</LDAPEntries>
```

Oracle Identity Manager Integration

The *Oracle Identity Manager* product allows a site to centralize their user definitions and password rules to manage and deploy across the enterprise set of products. When an employee joins an organization, changes their name or departs an organization their security presence across an enterprise must be appropriately managed. Oracle Identity Manager allows for users to be provision and managed in a central location.

An integration is provided to allow the ability to create, maintain and remove users in the identity management product and sync those changes to the users defined in the application. The following sections provide additional details about the integration with respect to configuration steps required in an Oracle Utilities Application Framework based product. For more information about the configuration required in the identity management product, refer to the *Identity Management Suite Integration* technical reference paper.

In order to use this functionality, feature configuration options for the **External Messages** feature type must be configured.

- Set option type **Support SPML Deployment in IWS** to **true**.
- Set option type **Default SPML service security policy** to an appropriate value per your implementation rules.

Template User Functionality

The user object in this product captures configuration used to control access but also preferences. The identify management product allows for extending the configuration to capture user configuration that is specific to this product. However, it does not support providing searches or dropdowns to select valid values. For example, to define the user's Home Page requires the reference to a navigation option. To set up your business process such that the home page is configured when defining the user in the identity management product dictates that the security user types in the correct navigation option reference.

On the other hand, to define a minimal amount of user information in the identity management product may result in a two step process for defining users: first define them in the identity management product with the basic authentication details and setting system defaults for some important fields, then after submitting the new user to be added to this product, navigate to the user page in this product and fill in all the configuration that is specific to this product.

The product provides support for defining a template that can facilitate the definition of users and reduce some of the challenges listed above. Refer to Template User for information about defining template users along with other details around this functionality.

Once you have template users defined, you should look at extending the configuration in the identity manager product.

- Map the information that is unique to a user and in addition, define a field for the template user. For example, you may choose to only capture the Name (first and last), Email address and User IDs for the user along with its Template User (which is mapped to a user characteristic). Additional fields may be included for capture in the identity management product when defining new users as per an implementation's business needs. For example, if the organization covers multiple time zones, perhaps it is easier to define the user's time zone when defining the user in the identity management product.
- When the new user is uploaded to the system, the algorithm used by the system to create the user looks for the existence of a template user sent as a characteristic of type **F1-TMUSR**.
- Once the new user is created, its configuration can now be adjusted, if applicable.

Note: There is configuration needed in Oracle Identity Management to capture the template user and any other information that the implementation has chosen to define in the identity management product when provisioning a new user. Refer to the *Identity Management Suite Integration* technical reference paper for more information.

Copyright © 2024, Oracle and/or its affiliates.

Provision Users with SCIM Open Standard

The product provides a REST web service **F1-SCIMUser** (SCIM User Provisioning) that supports adding, changing or viewing details of a user where the API follows the SCIM standard.

The following points highlight some of the mapping between the SCIM API and the application's user record.

- The user record in the application supports only first name and last name. The SCIM standard supports additional detail such as middle name, suffix and prefix. These elements are ignored when adding or updating a user.
- The SCIM standard supports a collection of email addresses. The application only supports one email address. As such, only the first email address is used when adding or updating a user.
- The integration relies on the use of a Template User so that the user is created with appropriate application configuration associated with each user. The system copies application settings from the template user to the new user being provisioned. If the external system is able to support referencing a template user such that it can be provided with the payload, the integration expects the user type element in the API to reference the template user. If it's not possible to provide a template user ID in the user type element, you should define a default template user using an algorithm parameter. Refer to Template User for more information.

Refer to the web service definition for more information about the supported integration.

Template User Functionality

The user object in this product captures configuration used to control access but also preferences. The identify management product allows for extending the configuration to capture user configuration that is specific to this product. However, it does not support providing searches or dropdowns to select valid values. For example, to define the user's Home Page requires the reference to a navigation option. To set up your business process such that the home page is configured when defining the user in the identity management product dictates that the security user types in the correct navigation option reference.

On the other hand, to define a minimal amount of user information in the identity management product may result in a two step process for defining users: first define them in the identity management product with the basic authentication details and setting system defaults for some important fields, then after submitting the new user to be added to this product, navigate to the user page in this product and fill in all the configuration that is specific to this product.

The product provides support for defining a template that can facilitate the definition of users and reduce some of the challenges listed above. Refer to Template User for information about defining template users along with other details around this functionality.

Once you have template users defined, you should look at extending the configuration in the identity manager product.

- Map the information that is unique to a user and in addition, define a field for the template user. For example, you may choose to only capture the Name (first and last), Email address and User IDs for the user along with its Template User (which is mapped to a user characteristic). Additional fields may be included for capture in the identity management product when defining new users as per an implementation's business needs. For example, if the organization covers multiple time zones, perhaps it is easier to define the user's time zone when defining the user in the identity management product.
- When the new user is uploaded to the system, the algorithm used by the system to create the user looks for the existence of a template user sent as a characteristic of type **F1-TMUSR**.
- Once the new user is created, its configuration can now be adjusted, if applicable.

Note: There is configuration needed in Oracle Identity Management to capture the template user and any other information that the implementation has chosen to define in the identity management product when provisioning a new user. Refer to the *Identity Management Suite Integration* technical reference paper for more information.

Copyright © 2024, Oracle and/or its affiliates.

User Redirection

The product allows an implementation to support the ability for a customer to authorize a customer service representative (CSR) to view the customer's account information in a customer facing portal (Digital Self Service), to see exactly what the end customer is seeing/experiencing.

When the request for the CSR to view the customer information is received, the following steps occur:

- The security server for the customer facing portal is contacted to obtain a temporary grant code.
- The redirection response returns the grant code and the URL of the target customer facing portal.
- The browser redirects to the user to this URL with the grant code, which is further authenticated and automatically exchanged for an authorization token, allowing the CSR to log in.

Standard security measures with private/public key encryption using JSON Web Tokens (JWT) ensures that only the implementation's enterprise application is allowed to communicate with the implementation's customer facing portal server. The system also provides a plug-in spot to allow for additional information, such as the customer's name and email address, to be included in the call to the security server.

This feature works in conjunction with functionality provided by the enterprise customer application. Refer to the documentation for that product for information about how this feature is supported.

Note: The product has been designed to potentially support other types of redirection besides the mentioned use case by adding another entry in the Redirection Target lookup (F1_REDIRECTION_TARGET_FLG). However, the internally provided services work specifically for the Digital Self Service use case. There is no guarantee that they would work for a different redirection use case. Please contact product support if your implementation would like to implement a different redirection scenario.

Redirection Security Context Plug-in

The product supports a plug-in spot to allow for the system to populate additional keys and values to add to the JWT (JSON Web Token) when calling the external security service.

In addition, the product provides an Extendable Lookup BO **Redirection Security Context (F1-RedirectionSecurityContext)** to define the key mnemonic values. For an enterprise product or implementation to populate additional keys, use the extendable lookup to define the key mnemonics. Then provide an algorithm that uses the available information about the customer and populates the security context keys and values by determining the related data in the system.

Master Configuration

The master configuration **Redirection Targets Configuration (F1-RedirectionTargetsConfig)** defines the information needed to contact the security server to obtain a grant code. It also defines the redirection URL.

Build an Anchor Hyperlink

The service script **Return a Hyperlink to the Redirect Servlet for a given Target (F1TargetLink)** is provided to allow for a link to be placed on a UI map or in a zone to initiate the redirection request. Refer to the service script detailed description in the metadata for more information.

Batch Scheduler Integration

The Oracle Database includes an enterprise wide scheduler to simplify the scheduling of background processes. The scheduler is implemented by the *DBMS_SCHEDULER* package. The product provides an integration with the Oracle Scheduler to facilitate scheduling background processes shipped with the product.

At a high level, the integration with the Oracle DBMS Scheduler supports the following entities:

Copyright © 2024, Oracle and/or its affiliates.

- **DBMS Program.** A program should be defined for each Batch Control that needs to be scheduled by the DBMS scheduler. A program would typically invoke a batch job, but it could be configured to set certain options instead.
- **DBMS Chain.** A Chain defines a series of steps with dependency rules between them. A step references a program, with the program performing the actual work for that step. A rule is attached to each step to identify its dependent steps and the condition for when that step should be executed. For example, in a chain consisting of STEP_A and STEP_B, where STEP_B can only start if STEP_A was successful, the rule for STEP_B to start would specify a condition of "STEP_A SUCCEEDED".
- **DBMS Schedule.** A predefined frequency for jobs that need to be run periodically, for example, nightly jobs.
- **DBMS Job.** Defines a plan to perform a specific program or a chain periodically on a specific schedule or ad-hoc.

The product provides a set of business services to maintain these entities as well as submit jobs, manage submissions and report on past submissions. Refer to business services that start with the prefix "F1-DBMS" for more information.

Note: For details on the integration, refer to the *Server Administration Guide* which contains the API. In addition, refer to the technical reference paper *Oracle Scheduler Integration* that provides guidelines for using this integration.

Data Synchronization

Your implementation may need to communicate certain data to external systems. This may be part of a data warehousing requirement or an integration effort. The synchronization process has two main parts. First, the change to the data must be detected and captured. Once that is accomplished, the next step is to manage the communication of that change to the external systems involved. The changes must be captured in chronological order so as to avoid systems going out of sync.

Understanding Data Synchronization

The product supports various options for managing an ongoing synchronization of an entity to an external system. While each integration involves unique business rules, in many cases the process is quite similar and involves the following overall steps:

- When an entity is added, changed or deleted logic determines whether the change is eligible for synchronization and if so a sync request record is initiated to manage the synchronization process. The base product uses the **Audit** plug-in spot on the maintenance object to allow for such logic to be performed when a change is detected to a record for that MO. The framework calls the algorithm defined on this plug-in spot in the event a change to the MO has been detected. Refer to the description of the plug-in spot on Maintenance Object - Algorithms for more information about when this plug-in is called.
- The processing of the sync request is deferred to batch so that the communication with the external system does not impact the updating transaction. It also ensures that the data being shared with the external system is fully committed to the database.
- When processed, sync request logic manages the following aspects of the process:
  - Determines whether the synchronization is still needed and if not discards the request.
  - Prepares the outbound message with the entity's synchronization details and sends it to the external system.
  - Manages errors and retry attempts as needed.

The following sections describe the various data synchronization options supported by the product.

Accumulative Data Synchronization

The product supports a data synchronization method by which several changes made to an entity may be accumulated and sent together as a single message to the external system.

The following points describe the functionality at a high level:

Copyright © 2024, Oracle and/or its affiliates.

- When an entity is added, changed or deleted, a snapshot of the entity's data from before the change is captured on one or more sync requests, one for each external system the data needs to be interfaced to. The before-update snapshot is used as a base line for detecting all changes made since then at the time of sending the message to the external system.
- Sync requests are processed by the next monitor batch process.
- The processing of the sync request typically involves business rules that compare the before-update snapshot with the current content of the entity to detect all change made. If changes were made, logic may decide to proceed with the sending the synchronization message otherwise the request is discarded. These rules are also responsible for handling any errors encountered while communicating with the external system and manage retry attempts as needed.

This functionality is supported by logic supplied in the base business object Sync Request (**F1-SyncRequest**). Note that each edge application delivers an appropriate child business object for this BO for each specific synch scenario supported in that product. Some of the functionality below is accomplished using configuration on the parent BO delivered by the framework while other functionality may be delivered by the child BO. In addition, there may be more complex use cases supported by your specific product integration. Refer to your specific application's library of Sync Request business object along with the documentation related to your specific product integration for more information.

The following sections describe key aspects of this data synchronization approach.

Capturing the Change

The base product provides a change data capture (MO Audit) algorithm **F1-GCHG-CDCP** that may be used by maintenance objects that need to be synchronized by this method. This algorithm creates a Sync Request record for the changed record, capturing the MO code and the primary key, if it doesn't find an existing sync request for the same record (and the same business object) in the initial state. The sync request business object used is the one defined in the **Sync Request BO** option on the MO for the record that was changed. If more than one such option exists then multiple sync requests are created.

Your specific product may also introduce additional Audit algorithms to cater for more sophisticated examples.

When creating the sync request record, typically the Sync Request BO will have a pre-processing plug-in that captures a snapshot of the record's data prior to its change. This will be used in subsequent steps to verify that the external system needs to be notified of the change.

Confirming that a Sync is Needed

Once a sync request is captured, there are several steps performed prior to any information being sent to the external system.

Note: This section only highlights key steps. Please refer to the business object configuration, its lifecycle and algorithms for a thorough picture of the full functionality..

- When a Sync Request record is created, its initial state (**Pending**) is configured to be processed by a batch monitor. That way, records are added to the sync request table throughout the day but all are processed together. The MO audit algorithm ensures that a new synch request is not created if a Pending record already exists for a given MO / PK combination (for the same business object). However, it is possible that a record for that MO / PK exists in a subsequent "non-final" (such as **Awaiting Acknowledgement**). This state includes a monitor algorithm to check for that condition and to skip transitioning if another record exists. This is done to ensure that the existing record is fully processed before this new record is processed.
- The next state of the lifecycle is **Determine if Sync Needed**. This step uses an algorithm to take a snapshot (called the 'final snapshot') of the data and compare it against the initial snapshot taken when the record was created. Based on the logic of the algorithm, it may decide to proceed (transition to **Send Request** or to discontinue (transition to **Discarded**.

Communicating to the External System

Once it is confirmed that the sync should occur, a message must be sent to the external system. The following points highlight the basic functionality.

Copyright © 2024, Oracle and/or its affiliates.

- An algorithm linked to the **Send Request** state. The expectation is that this algorithm creates an outbound message that routes the information to the external system appropriately. The algorithm must determine the external system and outbound message type to use. Business objects for Sync Request support BO options to define the external system and outbound message type to use for this algorithm.
- Once the outbound message is triggered, the record transitions to the **Awaiting Acknowledgement** status. This state is used to hold the sync request from further state transitions until an acknowledgement is received from the external system. Note that this step relies on implementation of a response mechanism from the external system. It is recommended to implement a response as this helps control the chronological flow of information. The product supplies the business service **F1-UpdateSyncRequest** that transitions the sync request to either the next default state (in this case the **Synchronized** state) if a positive acknowledgement is received; or the state associated with the Rejection transition condition (in this case the **Error** state) if a negative acknowledgement is received. In addition, this state may be configured with a monitor algorithm that detects that a timeout limit has been reached.
- For records that enter the **Error** state, it is recommended to configure an algorithm that creates a To Do entry to alert someone of the problem. Refer to the integration documentation for more information. The state is already preconfigured with an algorithm to complete To Dos when exiting the state.
- The final state **Synchronized** is used to mark the successful synchronizations. However, for more complicated use cases, this state may be used to trigger some additional action. Refer to the documentation for your specific product integration for more information.

Generic Data Synchronization

The product supports a generic data synchronization method by which each change to an eligible entity triggers a synchronization flow that communicates details about the entity to one or more external systems.

The following points describe the functionality at a high level:

- The integration of a specific entity (maintenance object) and an external system is enabled and configured using a **Data Synchronization Control** (F1-DataSyncControl) extendable lookup record.
- Rules supporting the integration reside in a **synchronization script** referenced on the control record.
- When an entity is added, changed or deleted, a sync request may be initiated for each integration according to business rules defined by the corresponding synchronization script.
- Sync requests are processed by the next monitor batch process.
- When the sync request is processed, data is synchronized to the external system according to business rules defined by the corresponding synchronization script.

This functionality is supported by logic supplied in the base business object Generic Data Sync Request (**F1-GenericDataSync**). This business object does not need to be extended to support specific integration rules as these rules reside in the synchronization script associated with each integration's data synchronization control record.

The following sections describe key aspects of this data synchronization approach.

Data Synchronization Control

The **Data Synchronization Control** (F1-DataSyncControl) extendable lookup defines the configuration needed to support an ongoing data synchronization of an entity to an external system. The configuration defines the entity's maintenance object, the external system it is synchronized with and references a synchronization script that manages the specific synchronization rules for this integration. An integration is considered enabled when its corresponding control record is in active status.

The data synchronization script is called in the following modes:

- When the entity is added, changed or deleted, the script is called in **Check** mode to determine whether a sync request should be created for the changed entity and a specific external system.
- When the sync request is processed, the script is called in **Process** mode to prepare and send the outbound message to the external system. Alternatively the script may request to discard the request or indicate that it should be processed later. The latter may be used when there is a need to wait until certain conditions are met before sending the message to the external system.

Copyright © 2024, Oracle and/or its affiliates.

Note that the configuration for a maintenance object and a specific external system may be overridden by another configuration defined with a higher sequence number.

Confirming that a Sync is Needed

The base product provides a change data capture (MO Audit) algorithm **F1ONGDATASYN** that may be used by maintenance objects that need to be synchronized by this method. The algorithm consults the synchronization script associated with each active integration control record configured for the entity as to whether the external system needs to be notified for this change. Once confirmed, the algorithm creates a Sync Request record for the changed record, capturing its MO code and the primary key, along with the corresponding external system.

Note that the decision as to whether to create a sync request or not is controlled by the integration specific synchronization script while the creation of the sync request is the responsibility of the audit algorithm. As such, the same sync request BO is used for all integrations, with no provision for capturing integration specific details.

It is recommended to design the synchronization script to only approve the creation of a synchronization request when it is needed. This prevents unnecessary creation of sync requests that would otherwise be discarded when eventually processed.

Managing the Synchronization Process

Once a sync request is captured, there are several steps performed prior to any information being sent to the external system.

Note: This section only highlights key steps. Please refer to the business object configuration, its lifecycle and algorithms for a thorough picture of the full functionality..

- When a Sync Request record is created, its initial state (**Pending**) is configured to be processed by a batch monitor. That way, the changes that triggered the request are fully committed before being communicated to an external system.
- When the sync request record is processed, a monitor algorithm calls the synchronization script associated with the corresponding integration's control record to manage the synchronization process. The synchronization script may indicate either of the following:
  - The request is not yet ready to be processed, i.e. it should remain in its current status and get processed again by the next batch run.
  - The request should be canceled. This indication results in the transition of the record to the **Canceled** status.
  - The request was successfully processed. The synchronization script has prepared the outbound message and routed the information to the external system appropriately. This indication results in the transition of the record to the **Sent** status.
- If an error occurs while processing the request, a To Do Entry is raised to capture the error details and the record is set to the **Error** status. Retry processing of the sync request continues with each batch process until the request is either sent or canceled.

Communicating to the External System

Typically the communication with an external system is finalized when the message is successfully sent and successfully processed by the external system. Depending on the specific integration, the message may not be processed right away by the external system but rather queued and processed later. In this situation, the sync request may need to be further acknowledged as successfully processed (set to the **Acknowledged** status) upon an explicit response mechanism from the external system. The product supplies the web service **F1-UpdateAndTransitionSyncRequest** that may be used to process such acknowledgements.

Maintaining Sync Requests

The system provides a Sync Request portal that is used to view the in progress or completed sync request records.

Copyright © 2024, Oracle and/or its affiliates.

The menu location of the portal depends on your specific edge product. It may be in a Data Synchronization menu or perhaps in the Batch menu. You are brought to a query portal with options for searching. The options may differ based on your specific product.

Once a sync request has been selected, you are brought to the maintenance portal to view and maintain the selected record.

An **Actions** zone may appear to display specific actions. Alternatively, the actions may be displayed directly in the display area of the **Sync Request** zone.

The **Sync Request** zone provides basic information about the sync request record.

Depending on your specific product additional zones may appear.

Operational Analytics

The framework provides several building blocks and tools that the edge applications may use to implement analytic reports directly from the operational application using Oracle Utilities Analytics Visualization product (referred to in this section as the analytics visualization product). The following sections provide more information about this functionality.

Understanding Operational Analytics

The following sections describe the type of configuration supported in your product to integrate with the analytics visualization product. Refer to the Oracle Utilities Analytics Visualization documentation for more information.

Direct Data Access

The analytics visualization product's canvases report directly off of the operational system (with the exception of batch run analytics) thus eliminating the need of an ETL process. Selected tables and views have been designated as dimensions and facts for the purpose of generating the Start Schemas used by these canvases.

Batch Run Analytics

There is a high volume of batch run data in the operational system that changes frequently and is recorded in a complex set of relationships. The product provides snapshot tables to capture only the batch run and thread data required for analytics. Using these tables for batch subject area views both improves performance and supports extracting the data for upload to other environments.

The snapshot tables are populated in increments. The **Batch Run Analytics Snapshot Update** and the **Batch Thread Analytics Snapshot Update** processes are run on an ongoing basis; each snapshot update selects only the batch runs and threads not yet included in the snapshot. Parameters are supplied to restrict the selected data based on a range of days from the current date, to support the initial uptake of data. It is recommended that batch run data is updated before the batch thread data, as only threads with corresponding records in the batch run snapshot will be added. Refer to batch controls **F1BRANSN** and **F1BTANSN** for more information.

The **Batch Run Analytics Data Export** and **Batch Thread Analytics Data Export** processes are provided to extract the batch data in increments for upload to other environments. These processes assume that a **Data Export Control** exists for each of the snapshot maintenance objects, to capture the date of the previous extract as a reference point for selecting the next set of records. Refer to batch controls **F1BRANEX** and **F1BTANEX** for more information.

Calendar and Time Dimensions

Analytic reports rely on calendar and time dimensions to support various hierarchical grouping by date and time.

The Calendar dimension provides a level-based definition of the standard calendar and fiscal calendar in a flattened representation that is commonly used in data warehouses. This is necessary to enabling customers to group by calendar week, months, quarter and fiscal calendar period, quarter and fiscal year etc. Note that fiscal information about a specific date is optional and sourced from your edge application specific accounting calendar

Copyright © 2024, Oracle and/or its affiliates.

In the same way, the Time dimension provides a level-based definition of each minute in a day, supporting reports that group by hours, AM/PM etc.

Note: Calendar dimension records are generated by a batch process; one record for each date in a specified period of time. You need to run the **Generate Calendar Dimension** batch process on an ongoing basis to cover the standard and accounting calendar days needed to support your business. Refer to **F1-BICDD** batch control for more information.

Note: Time dimension records are also generated by a batch process; one record for each minute in a day. You need to run the **Generate Time Dimension** batch process once to generate all the records. Refer to **F1-BITMD** batch control for more information.

Bucket Configuration

The analytics visualization product provides support for defining a set of ranges, each representing a bucket for which extracted measures can be grouped and classified under the relevant bucket. The framework product provides support for viewing and defining the buckets. Refer to Bucket Configuration for more information.

Analytics Attribute Mapping

The product supports mapping of characteristics to user defined fields associated with dimensions in the analytics visualization product. Each characteristic table that is associated with a dimension table is provided with pre-generated user defined fields identified by unique column sequence numbers in the analytics visualization product. The mapping of characteristics to dimension user defined fields is maintained directly on the dimension portal. You may also review these mappings using the Analytics Attribute Mapping portal.

Logical Dimension

The Element Name of a foreign key field to a dimension represents a distinct logical version of the dimension. This allows for different logical dimensions to be based on the same dimension table. For example, a "Location" dimension may be referenced as a "Main Location" on one fact and also as an "Alternate Location" on the same or other fact. While both elements refer to the same dimension as foreign keys, each represent a distinct logical dimension via their element names. In the same way, while each date field is implicitly considered as a foreign key to the Calendar dimension, each distinct element name establishes a unique logical dimension. For example, a "Start Date" element name and an "End Date" element name are distinct logical dimensions of the underlying Calendar dimension.

Some logical dimensions may represent an aggregated or summarized form of another logical dimension. They are also known as "shrunken" dimensions. For example, the "Calendar Month Dimension" view is an aggregated dimension of the Calendar dimension. Establishing a link between a summarized dimension and its detailed dimension allows the analytics visualization tool to drill from one fact to another for performance reasons when they are indirectly linked via the same logical dimension but at different aggregation levels. For example, "Fact A" represents daily records and "Fact B" aggregates the same measures at a monthly level. Associating the "month" field in "Fact B", which is a foreign key to the "Calendar Month Dimension" view, with the logical dimension name of the "date" field in "Fact A" allows analytic queries to use "Fact B" for queries at the monthly level or above which perform better than performing them on the more granular "Fact A".

Analytics Configuration

Depending on which product or products you have installed, there may be some configuration needed for the analytics visualization product. Refer to Defining Analytics Options for more information.

Calendar and Time Dimensions

This portal is used to view generated calendar and time dimension records.

Refer to Understanding Operational Analytics for an overview of operational analytic functionality.

Navigate using **Admin** > **Analytics** > **Calendar and Time Dimensions**.

Copyright © 2024, Oracle and/or its affiliates.

The following zones may appear as part of the portal's **Main** tab page:

- **Calendar Dimension.** This zone lists most recent calendar dimension records. You may filter the list to review records generated up to a specific date.
- **Calendar Date.** This zone provides information about a selected calendar date.
- **Time Dimension.** This zone lists all time dimension records.

Maintaining Bucket Configurations

Several key performance indicators in the analytics product look at measurement values (for example: the age of an asset or the age of debt) classified into a number of pre-defined groups also known as buckets. The overall metric can then be reported by the different buckets and allow various analyses.

For example, the age of an asset can be classified into the following buckets:

- Less than 6 Months
- 6-12 Months
- One Year and Older

The age of debt, also known as arrears can be classified onto the following buckets:

- 0-30 Days
- 30-60 Days
- 60-90 Days
- 90+ Days

The definition of the buckets is extracted to the Business Intelligence data warehouse, to be used as dimensions.

Bucket Definition Considerations

Each type of bucket is defined using a bucket configuration Business Object. The bucket definition considerations and/or rules will vary based on the bucket configuration business object used. The business objects available are driven by your specific product. For a list of available bucket configurations business objects, navigate to the business object page and view the business objects for the Bucket Configuration maintenance object.

Setting Up Bucket Configurations

To maintain the bucket ranges for the bucket configuration(s) applicable to your product, open **Admin** > **Analytics Configuration** > **Bucket Configuration.**

You are taken to the query portal where you can search for an existing bucket configuration. Once a record is selected, you are brought to the maintenance portal to view and maintain the selected record.

Note: Your specific product may also include an Analytics Configuration portal that displays the list of existing and potential bucket configuration records, allowing you to drill into this page to view the record in detail.

The **Bucket Configuration** zone provides basic information about the bucket configuration.

For more information about the elements supported refer to the zone's help or to the relevant analytics integration documentation for your product.

Defining Analytics Options

Refer to Understanding Operational Analytics for an overview of the operational analytics functionality.

Your specific product may require some system wide configuration to be defined when using operational analytics.

Navigate to the Extendable Lookup portal. Search for and select the **Analytics Options** business object.

Use the embedded help to review information about the options to define.

Copyright © 2024, Oracle and/or its affiliates.

Defining Analytics Tables

This portal is used to search for Analytics Table definitions. An Analytics Table record exists for each table or view in the operational system that represent dimension and fact tables in the Operational Analytics product.

Refer to Understanding Operational Analytics for more information.

You can access the portal from the**Admin Menu** > **Analytics Configuration** > **Analytics Table.** You are brought to a query portal with options for searching for a specific record. Once a record has been selected you are brought to its maintenance portal to view and maintain the selected record.

Defining Analytics Dimensions

This portal is used to maintain metadata for Analytics Dimension tables. A dimension record exists for each table in the operational system that represents a dimension table in the Operational Analytics product.

Refer to Understanding Operational Analytics for more information.

You can access the portal from the**Admin Menu** > **Analytics Configuration** > **Analytics Table.** You are brought to a query portal with options for searching for a specific record. Once a dimension record has been selected you are brought to this portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page

- **Analytics Table.** This zone provides information about the selected dimension record.
- **Fields.** This zone lists all the fields that define the dimension table. Fields may be selected from the dimension's underlying operation table using the **Excluded Fields** zone or added as computed fields.
- **Dimensions.** This zone lists dimension tables that this dimension references via foreign key constraints.
- **Referenced By.** This zone lists dimension and fact tables that reference this dimension via foreign key constraints.
- **Characteristic Mapping.** This zone manages characteristic mapping extensions for the dimension. Refer to analytics attribute mapping for more information.
- **Excluded Fields.** This zone lists fields from the dimension's operational table that are not included in the dimension table.

Note: The ability to introduce custom analytics dimensions is currently not supported.

Defining Analytics Facts

This portal is used to maintain metadata for Analytics Fact tables. A fact record exists for each table or view in the operational system that represents a dimension table in the Operational Analytics product.

Refer to Understanding Operational Analytics for more information.

You can access the portal from the**Admin Menu** > **Analytics Configuration** > **Analytics Table.** You are brought to a query portal with options for searching for a specific record. Once a fact record has been selected you are brought to this portal to view and maintain the selected record.

The following zones may appear as part of the portal's **Main** tab page

- **Analytics Table.** This zone provides information about the selected fact record.
- **Fields.** This zone lists all the fields that define the fact table. Fields may be selected from the fact's underlying operation table or view using the **Excluded Fields** zone or added as computed fields.
- **Dimensions.** This zone lists dimension tables that this fact references via foreign key constraints.
- **Excluded Fields.** This zone lists fields from the fact's operational table or view that are not included in the fact table.

Note: The ability to introduce custom analytics facts is currently not supported.

Copyright © 2024, Oracle and/or its affiliates.

Defining Analytics Attribute Mapping

This portal maps extensions to user defined fields provided for tables associated with an analytics dimension table that capture well defined extension data. A common example of this type of data are characteritics.

These mappings may be directly maintained on the operational Analytics Dimension portal. Refer to Understanding Operational Analytics for more information.

If your system is integrated with the Oracle Utilities Analytics product refer to Understanding Analytics Integration for more information about ETL based characteristic mapping.

You can access the portal from the **Admin Menu** > **Analytics Configuration** > **Analytics Attribute Mapping.** You are brought to a query portal with options for searching for a specific mapping record. Once a record has been selected you are brought to the this portal to review the information for the selected record.

The **Analytics Attribute Mapping** zone on the portal's **Main** tab page provides information about the mapping record.

Note: For characteristic mapping, only **Adhoc** and **Pre-defined** characteristic types are supported.

Note: For the product provides support for identifier mapping for operational analytics in this release but only the metadata definition is supported. The support for these types of attributes in the operational analytics product is in a future release.

Analytics Integration

The framework provides several building blocks and tools that the edge applications may use to implement integration with the Oracle Utilities Analytics product (referred to in this section as the analytics product). The following sections provide more information about this functionality.

Understanding Analytics Integration

The following sections describe the type of configuration supported in your product to integrate with the analytics product. Refer to the Oracle Utilities Analytics documentation for more information.

Master Configuration

Edge applications that include an integration to the analytics product typically include a master configuration record that captures information needed for the extract, load and transformation step, such as extract parameters. These records are provided by the specific edge products and my be viewed and maintained on the master configuration portal.

Note that your specific edge application may deliver an **Analytics Configuration** portal that displays the information from the master configuration record along with other analytics related configuration.

Bucket Configuration

The analytics product provides support for defining a set of ranges, each representing a bucket for which extracted measures can be grouped and classified under the relevant bucket. The framework product provides support for viewing and defining the buckets. Refer to Bucket Configuration for more information.

ETL Based Characteristic Mapping

The product provides objects to allow mapping configuration of characteristics in the product to user defined fields on dimensions in the analytics product. The mapping is used by the extract process to populate target user defined columns with corresponding characteristic data.

Copyright © 2024, Oracle and/or its affiliates.

This type of mapping relies on configuration in the Allowed Target Dimensions extendable lookup. The extendable lookup is used to define each Target Table in the analytics product that has one or more user defined fields that may be populated with characteristic values. It also defines the valid characteristic entities that may act as the source for the characteristic data.

Note: The framework does not provide any values in this lookup, but edge products that support mapping provide values in this lookup. Please refer to your specific product's integration chapter and refer to the Oracle Utilities Analytics documentation for more information.

Refer to Defining Analytics Attribute Mapping for more information.

Change Data Capture Using Sync Request

Depending on the specific edge application and version you are using, there may be components of the integration that use Sync Request for the change data capture step. If that functionality applies to your implementation, the following points highlight how to get more information:

- Refer to the administration guide for Oracle Utilities Analytics to confirm if your product integration is using Sync Request for any change data capture functionality.
- Review the Sync Request Business Objects provided by your product for analytics integration.
- Refer to Data Synchronization for a high level understanding of the process.

Business Flags

It is possible that information detected in one product may be useful or even critical to share with another product. The framework provides functionality for receiving information from an external system that acts as a type of flag or alert that may need investigation. This allows any system to store detected business flags in a common way and share that information with one or more other systems.

Understanding Business Flags

The following is an example of a use case for business flags. Imagine that DataRaker highlights potential theft of service at a certain location. That product may initiate a business flag alert to various products owned by the implementation with a recognized "standard name" for the business flag, such as "TAMPER".

- If Oracle Utilities Meter Data Management receives this business flag, it may initiate a service investigation monitor.
- If Oracle Utilities Mobile Workforce Management receives this business flag, it may initiate a service investigation activity.
- If Oracle Utilities Customer Care and Billing receives this business flag, it may initiate a hold on billing for that location.

Note that the framework product supplies basic functionality to support logic that is common to all edge applications that implement business flag functionality. However it is the individual edge applications that supply more specific functionality (business objects and algorithms) for specific use cases, if applicable.

The following sections highlight functionality supported for business flags in the framework. Refer to the edge application product documentation for more details for supported use cases.

Standard Name

To ensure that Business Flags are universally understood across all edge applications and to simplify integration each Business Flag will have a Standard Name. This is a name that is used by all the products when sending information to each other. That way, if DataRaker product sends Oracle Utilities Meter Data Management a "TAMPER" business flag, it should result in the same functionality as when Utilities Customer Care and Billing sends a "TAMPER" business flag.

Copyright © 2024, Oracle and/or its affiliates.

Standard Names are defined using an extendable lookup. In addition to common extendable lookup fields, the standard name also references a category and a class. The lookup also supports defining one or more external names, for cases where information is communicated from an external system that does not send the expected Standard Name.

The framework delivers the class values but does not deliver any standard names or category values. Refer to your specific edge application products to verify if any standard names or categories are delivered. Your implementation may configure appropriate standard names and categories based on your business rules.

Business Flag Type Defines Behavior for a Standard Name

Although the definition of the business flag standard names should be universally understood by the various integrated products that support them, each individual product defines what should occur when a business flag with a given standard name is created. This is configured using a business flag type. Only one active business flag type may exist for a given standard name. Business flags that are received from an external system will define the standard name, but will not have knowledge of the specific business flag types defined. The business flag type is determined based on the standard name.

The business flag type defines the business object to use when creating the resulting business flag record. The business object defines the lifecycle of a resulting business flag record.

Business Flag Type Algorithms

The business flag type includes support for algorithms. This allows for an implementation to define a common business object that may be used for different business flag types (if a common lifecycle is followed) but allow for different functionality to kick in depending on the business flag type.

The product supplies a plug-in spot for **Additional Processing** that may be invoked by a business flag that enters the Additional Processing state. Refer to your product's library of business objects to determine if there is an Additional Processing state that supports calling algorithms on the business flag type.

Objects Linked to a Business Flag

There are two types of links between an object in the system and a given business flag.

Affected Entity

Each business flag is associated with a single record in the system that is considered the "affected entity" or the entity that the business flag is associated with. The affected entity is defined by the specific business objects designed for the use cases supported by your edge product. For example, many utility base products may configure service point as the affected entity for its business flag use cases. Each business flag created is linked to a specific service point. Linking a specific entity to each business flag allows for algorithms to trigger functionality for that entity such as an investigation order. In addition, algorithms may be implemented in other business process areas that look for the existence of a business flag and act accordingly.

Related Objects

The business flag supports linking one or more related objects to a business flag to make it easier to trigger functionality or for impacted business processes to look for business flags. For example, when creating a business flag for a given service point, it may be useful to link all the accounts that are currently linked to the service point. Then, if an account oriented process should check for a business flag, it can look directly for a business flag linked to the account in its related object.

Impacted Business Process

The product supplies support for associating one or more impacted business processes to a business flag type. This configuration is used when functionality for that business process is impacted in some way based on the existence of a business flag of a given type. For example, maybe some process is put on hold when a certain type of business flag exists.

Note that configuring a business process on business flag type is not enough to trigger any impact on that business process when a business flag exists. There must also be some logic implemented in the business process functionality itself that knows to look for a business flag for a given record that is configured to impact the business process.

Copyright © 2024, Oracle and/or its affiliates.

The definition of the business process is at the discretion of the edge application that supplies functionality to support this. For example, the business process could be defined as something broad such as “billing” or could be something more granular such as “billing estimation”. The system supplies an extendable lookup to use for configuring the supported business processes. Refer to the values of the business process extendable lookup in your edge application or to the edge application specific Business Flag documentation for more information about supported business processes.

Dates

A business flag supports two dates: a Business Flag Date/Time and a Business Flag **End** Date/Time

- The business flag date / time is required for all business flags. For some types of business flags only one date is needed.
- For business flags that have a start and end period, the business flag date/time acts as the start date and the other field is the end date.

For a business flag that has a date range, it may be important for functionality implemented for impacted business processes. How the process treats the date will depend on its functionality.

- For some processes, the business flag is essentially expired after the end date has passed. This applies to impacted processes that are only looking at the current status of data in the system. For example, collection processing could be held if there is a business flag currently in effect (where the current date is within the date range). It would never look at historical business flags.
- For some processes where historical data may be relevant, a business flag effective during that same historical period may impact the process. For example if a business flag denotes an outage event for a given time period, perhaps estimated consumption should never be calculated for that time period.

Note that because business flags have a status, the design for the lifecycle of the business objects for the above effective dated use cases must carefully consider the states. For business flags that are considered expired after the end date passes, the BO lifecycle may be designed to transition to a final state after that date such that the record is no longer included in active processing. For business flags that continue to impact processing for a historic period, the BO lifecycle may be designed to remain in a non-final state such that it is clear that the record is still applicable.

Creating Business Flags

Business flags may be created in a system for one of the following reasons:

- A message is received from an external system that initiates the creation of a business flag. In this case, logic in the external system has detected some situation that this product is being alerted about.
- Business logic in this product detects a situation that should be investigated or should act as a flag. In this scenario, there may not be any integration needed depending on the business rules.
- Business logic in this product detects a situation that another integrated product should be alerted about. In this scenario, the business flag record is used to send out information to the integrated product.
- A user manually creates a business flag based on knowledge of the affected entity. For example, a customer service representative may create a business flag as a result of contact with the customer.

Creating a Business Flag from a Web Service

The system supplies both a REST and SOAP inbound web service that may be used for an external system to initiate (or update) a business flag. The REST service is **F1-BusinessInsights**. The SOAP service is **F1-BusinessFlagSync**. In both cases, the web service invokes a "driver" service script whose ultimate responsibility is to determine the appropriate Business Flag Type based on the standard name or external standard name, and therefore the appropriate business object for the new business flag. Because different products may have different logic related to creating a business flag, the service script calls another service script linked to the maintenance object using the Business Flag Sync MO option.

The "driver" service script includes a "contextData" group, which in turn references the data area **F1-BusinessFlagContext**. This data area is to allow for an external system to provide additional information for a given business flag use case. For any integration that requires additional information to be passed in from the external system, a separate data area should be used to define the additional elements; and that data area should refer to **F1-BusinessFlagContext** as its extended data area.

Copyright © 2024, Oracle and/or its affiliates.

The framework does not supply a Business Flag Sync service script, however individual edge applications supply a service script based on the use cases it supports out of the box.

> **Note:** For products that are continuing to use XAI for external messages, the product also includes an XAI inbound service for the same Business Flag Sync Driver service script. Note that the product recommendation is to discontinue use of XAI and use inbound web services instead.

Error Handling

If there is a problem in trying to create a business flag based on incoming information, the Business Flag Sync Driver service script creates a special business flag record using the Business Flag Error Business Object. This is also configured on the maintenance object as an option. The framework product supplies the business object Business Flag Error (**F1-BusinessFlagError**) for this functionality. Refer to the business object description and configuration for more information.

Confidence

There may be use cases where a condition is suspected, but not confirmed. The originating system should be able to assign a "confidence" level to the business flag.

For example, Oracle Utilities Analytics Insights or other machine learning platforms will aggregate and analyze large amounts of data to identify potential insights that need investigation. These platforms can initiate a business flag with a **Suspected** confidence.

An application receiving this business flag may adjust the confidence to either **Confirmed** or **Rejected**.

- - Setting the confidence to **Confirmed** is the equivalent of a true positive.
  - Setting the confidence to **Rejected** is the equivalent of a false positive.

Note that the application that receives a business flag is responsible for acting on the value based on business rules. Any updates to the confidence can be communicated to the other products that received the same business flag, if desired.

Additionally, creating a business flag in the application with a confidence of **Missed** provides a way to denote a missed insight. This is the equivalent of a false negative.

Because a utility implementation may have multiple applications installed that support business flags, the following guidelines are suggested for designing where the confidence flag should be updated.

- If Oracle Utilities Service Order Management is implemented, it has the responsibility of updating the confidence flag and communicating the update to other products.
- Otherwise, the assumption is that Oracle Utilities Customer Care and Billing owns field work orchestration and that it will have the responsibility for updating the confidence flag and communicating the update to other products.

No product logic is provided to enforce the above suggestions, however, the business objects supplied by the different edge applications will support the recommended implementation.

Business Flag Updates from External System

When the product that is responsible for updating the Confidence flag makes a change, it should initiate an outbound message to alert other products. On the receiving side, the same inbound web service and Business Flag Sync service script is responsible for the update. Refer to Creating Business Flags for more information.

Setting Up Business Flag Configuration

The following topics highlight the general configuration steps required to use business flag functionality. Your specific product may supply additional functionality to support specific use cases for business flags. Refer to your specific product's documentation and the library of business objects supplied for Business Flag in your product for more information.

Copyright © 2024, Oracle and/or its affiliates.

Standard Name Category Characteristic Type

Define one or more categories for grouping your standard names into logical business groupings.

Navigate to the Characteristic Type page. Search for and select the Business Flag Category characteristic type (**F1–BUSFC**).

Define desired category values and descriptions to be used for the standard names.

Standard Name Lookup

Navigate to the Extendable Lookup portal. Search for and select the **External Insights Standard Name** business object.

Define values that are recognized in the external systems that your implementation is receiving business flag details from.

Define a **Category** for the standard name that is appropriate for your product. Note that the category does not have to be in sync with standard name definitions in external products.

Select a **Class** for the standard name that is appropriate for its usage.

Refer to the embedded help for more information about configuring this object.

Business Process Lookup

If your specific product supports configuring business processes that may be impacted by the existence of a business flag, they are defined as an extendable lookup.

Navigate to the Extendable Lookup portal. Search for and select the **Business Process** business object.

Integration Configuration

The following points highlight configuration required to support receiving business flag information from an external source:

- Define a record for each External System that the product may be receiving business flag records from. This should be a value known by the external system and provided when new business flags are sent to this product.

When this product should initiate business flag information to be sent to an external system, configure one or more Outbound Message Type records. For each one, update the External System to configure how each outbound message type is communicated to the external system.

Defining Business Flag Types

Refer to About Business Flags for an overview of business flag functionality.

To maintain the business flag types applicable to your product, open **Admin** > **Integration** > **Business Flag Type.**

This is a standard All-in-One portal.

The information captured on the business flag type depends on the business objects supported by your product. Refer to the embedded help text for more information.

Maintaining Business Flags

This section describes the functionality supported for viewing and maintaining business flags.

Refer to About Business Flags for an overview of business flag functionality.

Navigate using **Main** > **Integration** > **Business Flags**. You are brought to a query portal with options for searching for business flags.

Once a business flag record has been selected, you are brought to the maintenance portal to view and maintain the selected record.

Copyright © 2024, Oracle and/or its affiliates.

The **Business Flag** zone provides basic information about a business flag. Refer to the embedded help for more information.

Copyright © 2024, Oracle and/or its affiliates.

Chapter 12

Market Transaction Management

Topics:

- Understanding Market Transaction Management
- Configuring Market Transaction Management
- Maintaining Market Transactions

This section describes various entities and aspects involved in the configuration and maintenance of market transactions. The framework product supplies basic functionality to support the objects related to market transaction management. However more specific functionality (business objects and algorithms) for specific use cases are supplied by the edge applications.

Understanding Market Transaction Management

Several objects are provided to support market transaction management.

- Market Message Type. This object is used to define the different types of market messages, both inbound and outbound and to define the appropriate market message Business Object to use when creating objects of this type.
- Inbound Market Message. This object captures the instance of a market message received from an external source.
- Outbound Market Message. This object captures the instance of a market message that will initiate a message to an external source.
- Market Process Type. This object is used to define configuration related to processes that may be initiated to support steps that should be execute to support an inbound or an outbound market message. It defines the market process Business Object to use when creating market processes of this type. It may also define one or more market process event business objects to initiate events that are generated to support steps within the market process.
- Market Process. This object captures the instance of a market process that may be created to carry out business logic related to a specific inbound or an outbound market message.
- Market Process Event. This object is used to orchestrate individual events that may occur to support steps in a given market process.
- Market Configuration. This object may be used to capture configuration needed to support market transactions. Each type of configuration should use a Business Object to define what information needs to be captured. Each type of configuration may define up to five key fields that may be used to uniquely identify an individual configuration instance.

Configuring Market Transaction Management

The following sections describe market transaction management configuration options.

Defining Market Configurations

Each market configuration is used to define one or more options for a given type of configuration (based on a delivered business object).

The product defines a generic set of portals, described here, that may be used to define any type of market configuration. However, it may be that your implementation includes more specific configuration portals for certain types of market configurations.

Open this page using **Admin** > **Market Transaction Management** > **Market Configuration.**

You are brought to the **Market Configuration Query** where you need to search for the market configuration (i.e., its business object).

Once you have found the appropriate market configuration, select the value and you are brought to a standard All-in-One portal that lists the existing values for the market configuration. The standard actions for an All-in-One portal are available here.

Defining Market Message Types

Refer to Understanding Market Transaction Management for an overview of market messaging functionality.

Open this page using **Admin** > **Market Transaction Management** > **Market Message Type.**

This is a standard All-in-One portal.

The information captured on the market message type depends on the business objects supported by your product. Refer to the embedded help text for more information.

Copyright © 2024, Oracle and/or its affiliates.

Defining Market Process Types

This page is used to define market process types which control behavior of the resulting market processes and the valid market process events that could be triggered as a market process progresses through its lifecycle.

Refer to Understanding Market Transaction Management for an overview of market messaging functionality.

Open this page using **Admin** > **Market Transaction Management** > **Market Process Type.**

This is a standard All-in-One portal.

The information captured on the market process type depends on the business objects supported by your product. Refer to the embedded help text for more information.

Maintaining Market Transactions

The following sections describe various pages available to view and work on market transactions.

Maintaining Market Processes

Refer to Understanding Market Transaction Management for an overview of market messaging functionality.

Open this page using **Main** > **Market Transaction Management** > **Market Process.**You are brought to a query portal with options for searching for market processes.

Once a market process record has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The **Market Process** zone provides basic information about a market process. Refer to the embedded help for more information.

Related Objects

The related object tab includes several zones that display various objects that may be related to the market process.

Related Market Messages and Processes

This zone displays any entries in the related object collection that are outbound market messages, inbound market messages or market processes.

Related Processes

This zone displays market processes that are linked to this market process using the explicit related process table. The market processes linked here are marked either as a parent, a child or a sibling market process.

Other Related Objects

This zone displays market process events linked to this market process. In addition, the zone displays any miscellaneous objects that are linked to the market process in the related objects collection, besides outbound market messages, inbound market messages and market processes.

Maintaining Market Process Events

Refer to Understanding Market Transaction Management for an overview of market messaging functionality.

You can view this page when drilling into a market process event from the Market Process related objects tab.

Once a market process record has been selected, you are brought to the maintenance portal to view and maintain the selected record.

Copyright © 2024, Oracle and/or its affiliates.

The **Market Process Event** zone provides basic information about a market process. Refer to the embedded help for more information.

Related Objects

The related object tab includes a zone that displays all entries in the market process event's related object collection.

Maintaining Market Messages

Refer to Understanding Market Transaction Management for an overview of market messaging functionality.

Open this page using **Main** > **Market Transaction Management** > **Market Message.**You are brought to a query portal with options for searching for market messages. Both inbound market messages and outbound market messages may be queried here.

Once a market message record has been selected, you are brought to the appropriate maintenance portal (inbound market message or outbound market message) to view and maintain the selected record. The inbound market message portal and the outbound market message portal have similar portal tabs and zones.

The **Market Message** zone provides basic information about the market message. Refer to the embedded help for more information.

Related Objects

The related object tab includes several zones that display various objects that may be related to the market process.

Related Market Messages and Processes

This zone displays any entries in the related object collection that are outbound market messages, inbound market messages or market processes.

Other Related Objects

This zone displays any miscellaneous objects that are linked to the inbound / outbound market message in the related objects collection, besides outbound market messages, inbound market messages and market processes.

Chapter 13

Content Migration Assistant (CMA)

Topics:

- Understanding CMA
- CMA Configuration
- The CMA Execution Process
- Running Batch Jobs
- CMA Reference

This chapter describes the Content Migration Assistant (CMA), a facility to enable the export of data from one environment to another.

CAUTION: This chapter is intended for users responsible for creating environments for testing.

Understanding CMA

The Content Migration Assistant (CMA) provides implementers with a flexible, extensible facility for migrating data from one environment to another (e.g., from a development environment to a production environment). Data is exported from the source system to a file. The file can then be checked in to a version control system for reuse, or can be immediately imported into the target system and applied.

Note: As used in this chapter, *source* systems are those on which export-related activities are conducted and *target* systems are those on which migration updates are to occur.

The following points highlight some common use cases for using this tool:

- Seeding a newly created environment with configuration data. For example, if you are creating test environments, you can import configuration data from a development environment. In this case, you will probably set up a "copy all configuration" type of migration. Another use case is if you want to troubleshoot an issue in production, you may want to copy all production configuration to a test environment.
- Updating an existing test environment with new configuration or updates to configuration from a development environment. In this case your set of objects will be more targeted and you are doing more of an incremental migration.
- Promoting tested configuration to your production environment.
- Migrating selected master and transaction entities from one environment to another for testing purposes.

Here are key aspects of the import process that apply to all cases:

- When records exist in both the target and the source, the CMA tool will fully **replace** the object with the version from the source. Special consideration should therefore be given to records whose primary key is system generated to ensure the correct entity is being replaced.
- While typically the source version of an object is applied directly onto the target environment, CMA does provide a way to Adjust Imported Data.
- The import process **does not handle deletion** of records in the target environment. When importing records from a source environment to a target environment, the import process for the migrated records is able to identify objects to add and objects to change. There is no mechanism for indicating that records in the target environment should be deleted. The absence of those records in the import is not enough because the migration may be only importing a subset to add or update. If data on the target system must be deleted, users must delete the records in the target accordingly. Note that CMA does orchestrate the deletion of child rows of an object as a result of a comparison. This is only applicable to child records that are owned by the implementation.

Configuration steps are mainly used to define the data to migrate at the source system but also involve tasks to be performed at the target environment. Note that the products provide base delivered configuration that may be used as is or used as a template for building more specific configuration for a given implementation. At a high level, configuration tasks involve the following entities:

- System wide settings are defined in the **Migration Assistant Configuration** master configuration record.
- Each type of record that may be copied requires a **Migration Plan**. The migration plan is used to identify the maintenance object (MO) for the record (using a business object) and allows for instructions to specify related records that may be included in the migration.
- A **Migration Request** is used to define the data to include in a given migration export.

Refer to CMA Configuration for more information.

Once you have a migration request defined to include desired data to export, the next steps are to export from the current environment and import to the target.

- The **Export** process includes all the steps needed to select records to be exported from the source environment and create the export file. For more information, refer to Exporting a Migration.

Copyright © 2024, Oracle and/or its affiliates.

- The file created by the export, which is a BINARY file, needs to be transferred from the export directory to the import directory. The transfer needs to be done in such a way as to preserve the file structure. Refer to Additional Migration Considerations for more information.
- The **Import** processes include all the steps needed to read an imported file, compare the data in the file to the data in the target, review the proposed changes and apply the updates. For more information, refer to Importing and Applying a Migration.

The topics in this section provide a general understanding of the Content Migration Assistant (CMA) tool.

Types of Migrations

The Content Migration Assistant is used for two general types of migrations: wholesale and targeted. The following topics provide some additional information about these concepts.

Wholesale Migrations

Wholesale migrations are used when migrating all the configuration and/or administrative data from one environment to another. For example, a wholesale migration might be used when migrating administrative data from a development or test environment to a production environment.

A wholesale migration may be comprised of one or more migration requests that in total include all the administrative data to move. Migration plans used in wholesale migrations may be designed to omit subordinate instructions related to explicit foreign keys that are identified through constraints as they are not needed, assuming that the data they are referring to will also be included in the migration.

With the ability to group migration requests, the expectation is that implementations follow these guidelines:

- Multiple migration requests using the Criteria-based or Entity List migration request classes are used to group information logically together to allow for more reuse.
- A Group migration request is used for the export. This allows for one data set to export and one data set on the import side, simplifying the process. Note that depending on the amount of data, this may be a large import set to process. An implementation may find it easier to create multiple migration requests that break down the process into several steps.

You should consider that the framework product provides base migration requests and your specific edge product may provide base migration requests as well that may or may not include framework migration plans. Using the product provided migration requests is beneficial with respect to maintenance. As features are added to the product (including new administrative maintenance objects), any impact on CMA should be included in the product owned migration requests. If your implementation introduces new custom administrative maintenance objects that should be included in CMA, then custom migration plans and a custom migration request should be added. Your implementation can build a Group migration request that includes the base migration request and your custom migration requests to have a consolidated export.

Note: Refer to Framework Provided Migration Objects for information about migration requests provided in the framework product. Refer to your specific product's documentation for information about addition base provided migration requests.

Targeted Migrations

A targeted migration refers to migrating a specific subset of data from one environment to another. Migration plans used in targeted migrations are designed to be self-sufficient and include all necessary subordinate instructions needed to ensure that the exported entity can be imported without encountering referential integrity errors. With this type of migrations, any relationship explicitly excluded from the migration plan assumes the related entity already exists in the target environment. Examples of targeted migrations include:

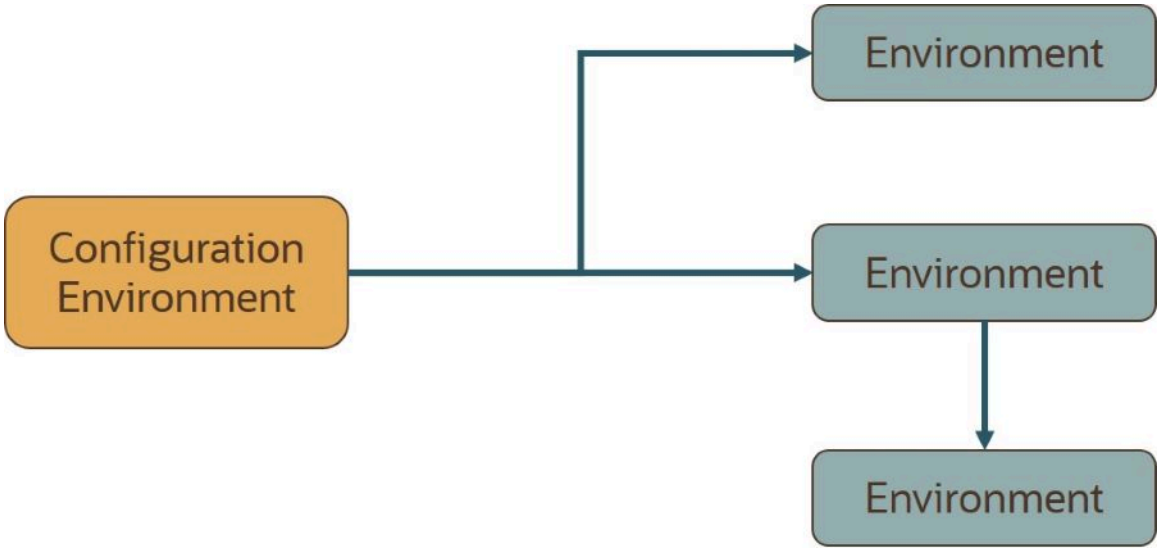
- Migration of a new portal, its zones, and its application service.
- Migration of all outbound message types.

Copyright © 2024, Oracle and/or its affiliates.

- Migration of selected accounts and their related master and transactional entities to a lower environment for testing purposes.

Migrating Configuration Data

A main use of CMA is to migrate configuration data, where the source environment is considered a kind of system of record and the target environment is getting updates based on changes to the source environment. Alternatively, you may be seeding a new environment. The expectation is that the target record is updated to look like the source record.

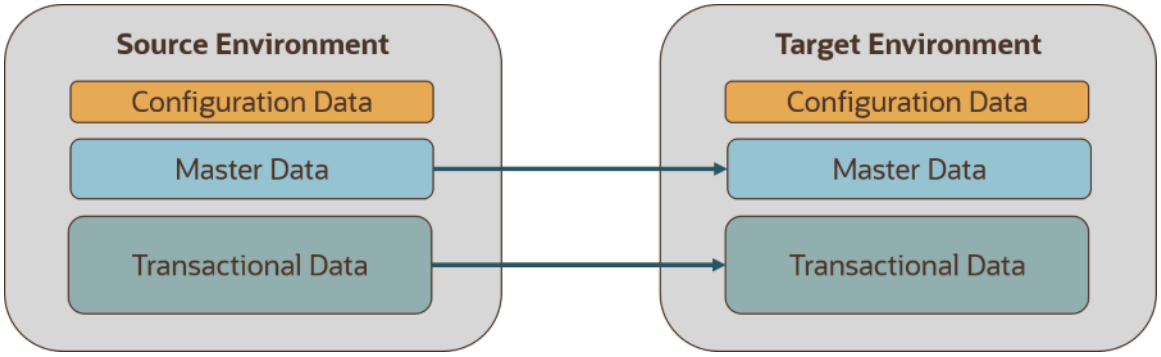


Configuration data may include a mixture of records that have an owner flag and ones that do not. The comparison step of the import process will generate appropriate insert or update SQL statements for the following data found in the export:

- Configuration data in a maintenance object with no owner flag. This is purely implementation data.
- Configuration data in a maintenance object with owner flag, where the owner is **Customer Modification**. For example, implementer-specific business objects.
- Configuration data in a maintenance object with owner flag, where the main record is owned by the product but where a child record exists with an owner of **Customer Modification**. For example, implementer-specific algorithms added to a base owned business object.
- Customizable fields in a record that is owned by the product. For example, the priority of a based owned To Do type.

Migrating Business Data

CMA may be used to perform a targeted migration of selected master entities and their related transactional data from one environment to another. For example, migrating a subset of accounts and their related data for testing purposes.



Copyright © 2024, Oracle and/or its affiliates.

The following points highlight key differences between business data and configuration data that call for special considerations when designing your business data migrations:

- Migration of business data typically involves a much higher number of records in a data set compared with a configuration-only data set. While there is no explicit limit to the size of a data set, some reasonable limit has to be assumed.
- Having large sets of entities depend on relatively few entities in the same data set significantly slows the import process. This situation occurs when mixing configuration data, master data and high volume transaction data in a single migration. We therefore recommend designing separate migrations for these data classes. Refer to the **Reduce Dependency Between Migration Objects** section below for more information.
- All business entities have a system generated key. Refer to Data with System Generated Primary Keys for more information.

The following sections expand on topics and concepts related to business data migration.

Reduce Dependency Between Migration Objects

CMA leverages the application's referential integrity and business rules to ensure entities remain compliant and valid when imported into another environment. These rules are there to prevent an entity from being imported when any of the entities it references do not exist in the target environment. When both the referencing entity and the entity it references are part of the same data set they establish a **dependency** that dictates the sequence in which these entities should be processed in order to be successfully imported. Note that the tool supports even the more complex and rare situations wherein the dependency between entities is cyclical, i.e. entity A (directly or indirectly) refers to entity B while entity B (directly or indirectly) refers back to entity A, for which sequential processing is not enough to guarantee a successful import.

The tool uses foreign key relationships between entities in the same data set to identify sets of dependent entities and group them into separate migration transaction records. Note that these foreign keys are defined in the application's data model repository and not at the database level for greater flexibility. The larger the migration transaction the more complex the process is to successfully import all its objects. So while the tool is designed to support all types of dependencies (sequential and cyclical), handling very large sets of dependent objects may incur a significant performance cost.

While configuration entities tend to be highly inter-dependent, the total number of entities that comprise the entire set of configuration objects is relatively small. Therefore, the impact on performance such dependencies have when importing a configuration-only data set is negligible. On the other hand, configuration entities tend to be referenced on many master and transaction entities as they control many aspects of the latter's business rules. As such, mixing configuration and business data in the same data set is likely to form extremely large sets of business entities that dependent on a small number of configuration entities. It is therefore strongly recommended to import or set up configuration data in the target environment prior to importing business data into it.

In a similar way, some transactional data tends to be of a much higher volume compared to the master data entities they reference. Here too, it is strongly recommend to migrate master data entities prior to their high volume related transactional data. For example, many interval measurement records (transaction data) reference the same measuring component (master data). Migrating a large volume of measurement data along with their measurement component would result in a very large migration transaction that may take long to import. Importing the measurement components prior to migrating their interval measurements will eliminate unnecessary dependencies within the measurements only data set and significantly improve the performance of the overall process.

Reasonable Data Volume

The overall volume of all business entities to migrate in a single data set should be reasonably sized. For example, importing several hundreds of accounts and their related master and transactional data is considered a reasonable size. Migrating too much data may reach physical and performance limitations of the tool.

You may use any supported migration request methods for describing the entities to export. If you opt for an Entity List migration request, consider using the Collect Entity zone to build the list of entities as you browse them on their respective portals.

Copyright © 2024, Oracle and/or its affiliates.

Single Source of Data

CMA uses an entity's prime key to determine if it is new to the target environment and therefore should be added or refers to an existing record and therefore needs to be replaced with the new version. All business entities have system generated keys, which are environment specific. It is therefore possible to have different entities in separate environments having the same prime key. When migrating entities whose keys are system generated, it is strongly recommended to migrate data to a target environment from a single source environment and avoid creating such entities in the target environment using the application. This practice ensures that prime keys for an imported entity are always in sync with their source environment. Refer to Data with System Generated Primary Keys for more information and considerations.

No Deletion

CMA does not handle deletion of entities of any kind, whether they are configuration or business entities. If a test needs to be repeated using an initial snapshot of the data, you should restore your target environment to a backup taken before test data was imported and import the latest version of the test data. If there is no need to delete previously imported entities, you may keep reloading test data from a single source environment as needed.

Bulk Import Mode

By default, the import process creates a migration object for each imported entity. This allows for granular reporting and error handling at the entity level. When importing a high volume data set of business entities, this granular management has a performance toll. In this situation, you may want to use the **Bulk Import** option on the migration data set import request. In this mode, a single migration object record is created for multiple entities of the same maintenance object. In the same way, a single migration transaction record is created for multiple logical transaction groups. Using this option reduces the migration object management effort throughout the process and thus results in better performance.

Note that each entity is still individually compared and validated, as in regular import processing, but if one entity is invalid the entire migration object is not applied hence impacting all entities grouped together with it. Bulk import mode is useful when importing a large set of data from a validated data source such that almost no errors are anticipated.

This option is only supported for master and transaction maintenance objects, i.e. it is not applicable to configuration migrations.

Insert Only Mode

By default, the comparison step of the import process needs to determine whether the imported entity is new and therefore should be added, represents a change to an existing entity and should be updated or the entity is unchanged. When importing a high volume data set of business entities that are all new to the current environment, this check is time consuming and can be avoided. In this situation, you may use the **Insert Only** option on the migration data set import request to indicate that all imported entities are assumed to be new additions to the current environment. By doing so, the import process avoids unnecessary steps to determine whether the entity should be added or updated and thus contributes to a faster import process.

This option is only supported for master and transaction maintenance objects, i.e. it is not applicable to configuration migrations.

Separate Batch Processes for Managing Business Migration Objects

By default, the same import related batch processes manage configuration and business data migrations. Typically, business data migrations involve high volume of records compared to much lighter configuration data sets. Processing them together by the same batch process may slow down the performance of configuration migrations, preventing them from completing faster and more frequently. The issue mainly affects test-like environments where mixed data class migrations are more common: configuration data is imported from a lower environment and large test data is imported from a higher environment. You may adjust the base product's configuration in such environments to benefit from separating the import processes for configuration and business data.

Note that the ability to separate import processes applies to migration objects only because of their volume. Migration data set and transaction records are of low volume and as such remain managed by the same batch processes.

Copyright © 2024, Oracle and/or its affiliates.

Designated batch processes are provided for importing migration objects containing business data but they are not used by default:

- F1-MGOPB - Migration Object Monitor (Business)
- F1-MGOAB - Migration Object Monitor (Business) - Apply

Follow these steps to use these separate designated batch processes for business data:

- Update the **Migration Object Business Data** (F1-MigrObjectBus) business object to reference the business data related batch controls on the following statuses:
  - **Pending, Error Applying, Needs Review** - Migration Object Monitor (Business) - F1-MGOPB
  - **Approved** - Migration Object Monitor (Business) - Apply - F1-MGOAB
- If your organization uses event driven job submissions for CMA batch processes then refer to Running Batch Jobs for additional configuration steps.

Data with System Generated Primary Keys

For records with system generated keys, there is an issue if the target environment already has a record with the same key but doesn't represent the same record. The tool shouldn't be just updating that target record with the view of the source in case this will break foreign key relationships that exist in the target.

The tool provides support for administrative data with system-generated primary keys. The logic relies on the maintenance object to use a method that looks at other attributes of the record (considered a "logical key") to detect whether the record being migrated already exists in the target region or not. The examples in this section are based on the Attachment maintenance object. Common attachments are considered administrative data. The attachment MO uses the file name and the creation date as the "logical key".

Imagine a common attachment for the "standard rate codes" file exists in a source region with the key 123456789. The table below highlights possible situations at the target region and actions supported in CMA.

Scenario	Target Situation	Action
1	No matching record	Record can be added with key 12345678
2	Record exists with key 123456789 and logic confirms that it is also the "standard rate codes" attachment.	Record can be updated.
3	Record exists with key 123456789, but logic detects that it is not the "standard rate codes" attachment.	Record is not updated. An error is issued
4	The system detects that another attachment record exists for the "standard rate codes" attachment with a different ID.	Record is not updated. An error is issued

The use cases described in scenarios 3 and 4 above would require key mapping to keep track of the id from the source to the id in the target so that any other records from the source that reference this key as a foreign key would be updated as part of the migration. This functionality is not supported.

Scenarios 1 and 2 above are supported for maintenance objects that use the method to detect the logical key.

Note: If a maintenance object with a system generated key does not supply a method to detect the logical key, CMA will update an existing record with the same ID. For maintenance objects in the framework that provide this method, refer to Framework Provided Migration Configuration. For your specific edge application, refer to the CMA addendum for information about support for data with system generated keys.

The product recommends that an implementation establishes a migration strategy such that records with system generated keys are always created in the same region and always follow a standard migration path for promoting the data from this source region to other regions. Following this strategy, you would minimize or eliminate the possibility that

Copyright © 2024, Oracle and/or its affiliates.

a record for the same logical key is created in multiple places such that different IDs would be generated as described by scenario 4 above.

MOs with a Mixture of Administration and Non-Administration Data

There are some MOs that contain a mixture of master or transaction data and administrative data. The Attachment is an example of this. The product supports common attachments and owned attachments. Owned attachments are records that are specific to its owner. The owner could be master or transaction data and its attachments are therefore considered master or transaction data. Owned attachments are not candidates for migration using CMA. Common attachments on the other hand are considered administrative data and may be candidates for migration using CMA. For these use cases, an implementation may follow the suggested strategy of only creating the administrative data in one region so that IDs for common attachments are not reused. However, it is reasonable and expected that owned attachments are being created in the target region and may receive a system generated key that matches the key of a common attachment from the source region.

To try to minimize this issue, the system includes special logic to be used by any MO that may contain administrative data mixed in with master or transaction data. This special logic generates the key of an administrative record with a zero (0) in the middle of the key and ensures that the keys for master and transaction data do not include a zero in this spot. For maintenance objects in the framework that use this method, refer to Framework Provided Migration Configuration. For your specific edge application, refer to the CMA addendum for information about additional maintenance objects that may be in this category.

Importing Data that References a User

Although the tool supports copying users from one environment to another, the assumption is that it will be more common for implementations to define their valid users for each environment separately. In fact, the default CMA configuration for copying security configuration assumes that users are not being copied.

There may be times were a record is being copied from one environment to another where a user is referenced on the record but that user is not valid in the target environment. For example, if a batch control is configured as Timed with a user referenced, if this batch control is copied to another environment, what should happen if that user is not found? You may not want that user defined in the target environment. To handle this, the product provides support for a Default User that you can define in the migration assistant configuration. If this default user is configured then any time the CMA import process recognizes that a record references a User and the user does not exist in the target environment, it updates the record to instead refer to the default user.

Note: Please note that if you are trying to copy entities that includes valid users, you should ensure that all the users are copied to the target environment first. Otherwise, the entity will be updated to refer to the default user, which may not be your intention.

Note: If you copy records that reference a User within an XML field or CLOB, the CMA input tool is not able to detect that. A pre-compare algorithm is recommended if this use case exists in your implementation.

Additional Considerations

The following sections describe some additional considerations for CMA.

File Transfer Considerations

When moving the export file between systems, use the binary transfer option of whatever tool you use to move the file so that line-end characters are not converted from Linux-style to Windows-style or vice versa.

It is recommended to avoid using 'txt' for the export file's extension (defined in the master configuration). That file extension by default implies a non-binary file and tools that perform file transfer may treat this as a non-binary file unless explicitly stated. The recommendation is to define 'cma' as the extension. This is not a recognized file extension and most file transfer tools will transfer the file as is.

Copyright © 2024, Oracle and/or its affiliates.

Note that if the file gets converted, there are two likely outcomes - either a numeric conversion error, or a buffer under-run error may be received when attempting to import the file.

Multi-Language Environment Considerations

If your implementation uses a language other than English, it means that migrated administration objects may have multiple language rows (because English is always enabled). There are some important points with respect to multiple languages and CMA:

- As described in User Language, there are steps to follow when supporting an additional language. The steps outlined in that topic highlight that for system data, translation of the strings may be provided via a language pack provided by the product or may be the responsibility of your implementation. In either case, this effort is non-trivial and will have its own established plan. The expectation is that the translation of the system data is applied for each region at your implementation site. CMA should not be used to create a new language in a target region.
- For administrative / control data that your implementation develops as part of your project, the expectation is that descriptions for your supported language are entered in the region that is considered the source region used to promote changes to regions in the "chain". For example, control data is entered in a development region and promoted to a test with the supported language enabled in both regions.
- What if you export data from a region with more languages enabled than your target? This scenario is perhaps a case where the source region is a type of test or playpen region where the additional language is enabled for other purposes. In this case, if the language code doesn't exist at all in the target region, the import will produce an error given that the code is invalid. If the language code exists but is not enabled, this will cause the extra language rows to be inserted in the target region, but will not cause any issues. They are simply ignored.
- What if you export data from a region with fewer languages enabled than your target? In this situation, the import process will only create language rows for the languages that were copied from the source. It will not automatically create language rows in the target as part of the import. For this situation, the recommendation is to run the **New Language** batch program (**F1-LANG**) that creates any missing language entries.

Versions and CMA

This topic addresses two separate version related topics with respect to using CMA between two environments.

- Different versions of the application
- Different CMA file format versions

Different Versions of the Product

CMA was developed with the main requirement of copying data between two environments that are on the same version of the product. For example, copying configuration data from a development environment to a test environment on the same product verison.

In general, there isn't anything about how CMA functions that would limit your ability to copy data between two different versions of the product. However, it's possible that your product has enhanced the functionality of a specific object to add or remove columns or change the type of data captured in those columns or has change validation around how data should be captured.

Our products follow strict standards for upgrading clients to ensure backward compatibility. Upgrade scripsts in the blueprint process know what was changed from one release to another and update the data accordinly where applicable. When using CMA to move data from one version of the product to another, you are bypassing these upgrade scripts. It means your data is not being adjusted as expected. It's not recommended unless you are certain that the data of an object from one version is compatible in the other version. The following points highlight this in more detail.

- For objects whose structure and rules have not changed between the two versions, there should be no issues.
- For objects that have changed between the two versions, there are scenarios that may cause additional steps.
    - If the target environment is a later version and includes a new optional column on an object that was not present in the source, the import should go through without any issues.

Copyright © 2024, Oracle and/or its affiliates.

- If the target environment is a later version and includes a new required column on an object that was not present in the source, a 'pre-compare' algorithm could be used to populate the new required value. Your implementation can provide this algorithm to populate the data according to your business rules.
- If the target environment is an earlier version than the source, we cannot guarantee that there won't be specific issues encountered with specific objects.

File Format Version

If your implementation has decided that you are going to use CMA across product versions, given all the cautions described in the previous section, this section provides information on the rare case where CMA has a change in who it writes the export file.

When creating the export file, the format of the file is specific and the CMA import step knows how to read the file. It is rare, but there are times when the product makes an enhancement to CMA that requires a change in the structure of the export file at which point the 'version' of the CMA file is incremented.

The CMA file format version has no impact to migrations that are exported and then imported into the same version of Oracle Utilities Application Framework. However, if your implementation has different environments on different versions of the product where the CMA file format is also different, then the system has functionality to cater for these differences.

- If you are exporting data from a lower file format version to a higher file format version, there is nothing special that you need to do. The export step captures the file format version and the import step reads this information and processes the file accordingly.
- If you are exporting data from a higher file format version and the product currenly supports exporting to a lower file format version, at export time you are prompted for the earlier file format version. You may indicate the you want the earlier file format version when creating the migration data export.

> **Note:** The product supports exporting to an earlier file format version for a reasonable amount of time, but not in perpetuity. If you don't see any option in the user interface to export to an earlier file format, it means that it is not supported in that release.

CMA Configuration

The following sections describe tasks required for CMA configuration.

Master Configuration - Migration Assistant

The **Migration Assistant Configuration** master configuration record defines system wide configuration used in both the source environment and the target environment.

As described in Importing Data that References a User, if you have records being imported that may reference a user does not exist (and shouldn't exist) in the target environment, define a Default User here.

In both the source environment and the target environment, the system needs to know the location of the export directory and the import directory along with the expected file suffix. For these fields, implementations may also rely on the system defaults. If no Migration Assistant Configuration record is found, the system assumes that there is an entry defined in the system's substitution variable list for **F1_CMA_FILES**. Further it defaults the values as follows:

- Export directory is the value for this variable plus **"\export"**.
- Import directory is the value of this variable plus **"\import"**.
- File suffix is set to **cma**

Refer to Referencing URIs for more information about the substitution variable list.

Copyright © 2024, Oracle and/or its affiliates.

For additional information about specific fields in the master configuration, refer to the embedded help.

Note: This record can be updated at any time to change details. The new configuration takes effect on all subsequent exports and imports.

Migration Plans

A migration plan defines one or more types of objects that are eligible for migration. It is essentially a set of instructions describing how the data to be exported is structured, allowing objects to be migrated together as a logical unit to ensure consistency and completeness.

The migration plan is always related to a given maintenance object, defined using its primary instruction. Multiple migration plans may exist for a given maintenance object if there are different requirements for migrating records in that MO under different circumstances. For each migration plan, you decide if additional data should be included as part of an object using subordinate instructions. For example, when copying a single algorithm type, should the system only copy that algorithm type record, or should it also copy all its algorithms and its related script (if applicable)? Refer to Types of Migrations for more information.

The following topics describe defining a migration plan as well as other topics for a migration plan.

Defining a Migration Plan

To view or define a migration plan, navigate using **Admin** > **Implementation Tools** > **Migration Plan.**

Use the **Migration Plan Query** portal to search for an existing migration plan. Once a migration plan is selected, you are brought to the maintenance portal to view and maintain the selected record.

CAUTION: Important! If you introduce a new migration plan, carefully consider its naming convention. Refer to System Data Naming Convention for more information.

The following points provide information about defining **Instructions** for a migration plan.

The **Instruction Sequence** uniquely identifies the instruction. The recommendation is to use increments of 10 to allow insertion of other instructions in the future.

Select **Primary** for the first **Instruction Type**. All migration plans must contain one and only one primary instruction. All subsequent instructions require a **Subordinate** instruction type. In this case, the **Parent Instruction Sequence** must be entered. This number, used to maintain the defined relationships in the exported data, must match an instruction sequence number at a higher level in the hierarchy.

The instruction **Description** provides a business description of the instruction.

Select a **Business Object** (BO) to define the type of object from which data will be derived.

Note: Though BOs are specified in each instruction, it's important to understand that each BO is used only for filtering purposes. The migrated data set comprises the complete contents of the *maintenance object* that the business object structure is defined against. For a more detailed explanation of this, see Understanding the BO Filtering Process.

Note: Refer to Identifying Tables to Exclude From Migrations for information about defining child tables to always exclude from a migration.

Traversal Criteria is used to define the relationship between each of the objects in a migration plan. The system provides three options to define how the child object is connected to the parent object so the system knows how to traverse from one object to another. **Traversal Criteria Type** options are **Constraint**, **SQL** and **XPath**. The following points explain each option:

- **Constraint** allows you to select a table constraint that represents a given record's relationship to another record in the system via a foreign key constraint defined in the meta-data. If **Constraint** is selected, the following additional fields are enabled:

Copyright © 2024, Oracle and/or its affiliates.

  - **Constraint ID** is a unique identifier for the constraint. The search will show the valid table constraints for the MO of the instruction's BO and the MO of the parent instruction's BO.
  - **Constraint Owner** is used to define the owner of the constraint. This is populated automatically when selecting a constraint from the search.
- **SQL** lets you specify SQL join criteria between the parent instruction's object and the child object in the **SQL Traversal Criteria**. The syntax of the the traversal criteria is a WHERE clause (without including the word WHERE). When referring to a field on the parent instruction's object, use the syntax `#PARENT.TABLE_NAME.FIELD_NAME`. When referring to a field on the current instruction's object, use the syntax `#THIS.TABLE_NAME.FIELD_NAME`. For example, the following statement is used on a migration plan for Business Object, where the parent instruction is the BO and the subordinate instruction is used to reference the UI Map that is referred to as a BO option with the option type "F1DU":`#PARENT.F1_BUS_OBJ_OPT.BUS_OBJ_OPT_FLG = 'F1DU' AND @trim(#THIS.F1_MAP.MAP_CD) = @trim(#PARENT.F1_BUS_OBJ_OPT.BUS_OBJ_OPT_VAL)`.
- The **XPath** option lets you apply syntax in an XPath expression referencing elements in the instructions' referenced business objects. This is entered in the **XPath Traversal Criteria**. For example, the display map collection statement in the SQL example noted above would be written as follows in XPath: `#this/mapCd = #parent/businessObjectOption/businessObjectOptionValue AND #parent/businessObjectOption/businessObjectOptionType = 'F1DU'`. This technique allows foreign key references that are mapped inside an XML column to be referenced.

> **Note:** The #parent expressions may access elements that are stored in an XML column and described using mapXML and mdField. However, the #this expressions must refer to fields available in the business object using the mapField reference.

Defining **Next Migration Plan** provides the ability to indicate that in addition to copying the object defined in the instruction, any additional instructions included in that referenced migration plan will also be included in an export.

The **Algorithms** grid contains algorithms associated with each instruction. You must define the following for each algorithm:

- Specify the **System Event** with which the algorithm is associated (see the table that follows for a description of all possible events).
- Specify the **Sequence** and **Algorithm** for each system event. You can set the **Sequence** to 10 unless you have a **System Event** that has multiple **Algorithms**. In this case, you need to tell the system the Sequence in which they should execute.

System Event	Optional / Required	Description
Pre-Compare	Optional	Algorithms of this type may be used to adjust the data after it is moved to the target system. These may only be defined on the primary instruction. Refer to Adjusting Imported Data for more information.
Import	Optional	Algorithms of this type are no longer supported.

Understanding the BO Filtering Process

Migration plan instructions require the definition of a business object to provide CMA with information about the record related to the instruction.

If the business object is the physical business object for the maintenance object, then CMA assumes that the instruction applies to all records that satisfy the traversal criteria. CMA recognizes the physical BO by comparing the BO to the value defined in the maintenance object option. If the business object defined is not the physical BO, then CMA will limit the records in the instruction to those that explicitly reference this BO or reference a child of this BO as its identifying BO value. (In other words, this BO must be in the parentage hierarchy of the records to be included in the instruction.)

Copyright © 2024, Oracle and/or its affiliates.

> **Note:** Unlike Bundling, CMA does not use the BO schema to drive what data is copied for a given record. The BO is only used as a filtering mechanism for selecting records. Refer to Identifying Tables to Exclude from Migration for information about how to ensure a child table is not included in a migration.

For example, if you define a migration plan for Master Configuration and use the physical business object for the instruction (**F1-MstCfgPhysicalBO**) then all master configuration records are considered for the instruction. If instead the business object you define is Migration Assistant Configuration (**F1-MigrationAssistantConfig**) then only the record related to this business object is included in the instruction.

Migration Plans for Objects with XML-Embedded Links

When migrating objects where foreign key references are captured in the object's XML based field, subordinate instructions are needed to define the foreign key references in order for CMA to understand the relationships. This is in contrast to direct foreign keys where CMA can determine the relationships using constraints. The instructions provide two purposes. Defining instructions ensures that the related objects are included in the migration, if appropriate. If you know that the related objects are included because of other criteria, the instructions allow CMA to group related objects into transactions. This helps in the apply process at import time to ensure that related objects are grouped together. However, the apply process includes iterative steps to try to overcome dependencies like this so defining the instructions is not critical for this purpose.

The following are options for creating migration plans with XML-embedded links:

- One option is to use the specific logical (business) BO in the primary instruction to define the object you are copying. With this option, the subordinate instructions may use XPath criteria to define the related foreign key. When this approach is used, a separate Migration Plan must be created for each logical BO. (Refer to Understanding the BO Filtering Process for more information.) This option would only be used in isolated cases.
- Another option is to create a migration plan that uses the Physical BO as the primary instruction, and then include a subordinate instruction for the real logical BO, using SQL Traversal to join the object to itself by its primary key. Note that with this technique, the records that reference the logical BO will still only be included in the export file once. At this point further subordinate instructions may use XPath notation to define the foreign key data. Using the physical BO as the primary instruction ensures that all records in the MO are considered. The subordinate instructions with the logical BO and XPath notations will only apply to the records that are applicable to that BO. This option is useful for MOs that have a small number of logical business objects with disparate foreign keys.
- Another option is to use the physical BO in the primary instruction and use raw SQLs in the subordinate instruction's traversal criteria to identify the foreign keys using substring commands. A separate Subordinate Instruction is needed for each SQL corresponding to each element occurrence. Using this technique has the same advantages of the previous in that all records for the MO are included in the migration. However, this technique may be useful for maintenance objects with a larger number of business objects expected where each has one or more foreign keys. It's especially useful if many business objects reference the same foreign key. Then only one instruction is required for that foreign key. Note that a single migration plan may use this technique and the XPath technique for different elements.

A migration request may have multiple migration plans for the same maintenance object. That allows for some flexibility and long term maintainability in that the above techniques may be used in multiple migration plans. Consider the following example:

- A product provides base business objects with foreign keys defined in the XML field and provides the appropriate migration plan with instructions. An implementation extends this business object or perhaps creates their own business object for the same maintenance object and includes different additional foreign keys in the XML. Rather than duplicating the base migration plan and adding additional instructions for the additional foreign keys, the implementation can create a second migration plan for the MO with the additional foreign keys defined. A migration request should be defined to include both migration plans. In this case if the implementation has only one custom BO, they can choose to use the custom BO as the primary instruction as described above in the first option.

Copyright © 2024, Oracle and/or its affiliates.

Defining a Migration Request

Migration Requests are used to define the data to be included in a migration. To view or define a migration request, navigate using **Admin** > **Implementation Tools** > **Migration Request**.

Use the **Migration Request Query** portal to search for an existing migration request. Once a migration plan is selected, you are brought to the maintenance portal to view and maintain the selected record.

There are three types or classes of migration request. The system provides a base business object for each along with a migration request class, which matches the business object. The subsequent sections provide more information about each class of migration request.

Note that all migration requests support defining a Category, which allows implementers to categorize the migration request.

In addition, all classes of migration request include the following zones:

- **Migration Request** This zone contains display-only information about the selected record. Please see the zone's help text for information about this zone's fields.
- **Referencing Migration Requests** This zone is only visible if the displayed migration request is included in a Group migration request. It lists each group migration request that includes it.

Other zones may appear for specific classes of migration requests. See the following sections for more information.

Criteria-based

This type of migration request defines a set of entities to be logically included together as part of a migration. For each migration instruction, selection criteria is defined to indicate which records for each MO should be included. Selection may be defined using SQL, an algorithm, an entity tag or explicit primary key values.

- For selection using SQL Statement, refer to the embedded help for examples.
- For selection using Algorithm, the algorithms that are eligible to be plugged in here are those valid for the **Migration Request - Select** system event.
- For selection using Entity Tag, reference a tag name. The entities linked to the tag at export time would be included in the export file.
- For selection using Specific Keys, the primary key (1 through 5) must be explicitly specified. Multiple rows are allowed.

Entity List

This type of migration request allows the user to choose explicit MO / prime keys. The MOs that are eligible are those that are configured with a **Default Migration Plan** option. Although the user is managing MO / PKs, the migration instructions are still defined with a migration plan. The system maps the migration instructions in a similar way to a **Criteria-based** migration requests that use a Specific Key selection type. Note however that it will create a separate migration instruction for each MO / PK combination. It does not try to group all PKs for the same MO / migration under one migration instruction.

For this type of migration request, a user adds a migration request record with its description and other main information. Then, a special zone **Add Entities** is provided to find and select records based on a selected maintenance object and add to the migration request. The user is prompted to provide a reference and comments, if desired. If the category selected is one that requires a reference, then this will be validated. Alternatively, you may also use the Collect Entity zone to add specific entities as you review them on their respective maintenance portals.

When maintaining a migration request with existing entities, they are visible in the zone **Migration Request Entities** . This zone allows a user to remove the entity from the list.

Copyright © 2024, Oracle and/or its affiliates.

Group

This type of migration request points to other migration requests. This allows you to define separate migration requests that represent logical groupings of migration plan instructions for ease of maintenance, but to combine all the separate migration requests into a single "grouped" migration request for streamlined export / import purposes.

The CMA export process will build an extract that includes the union of all the objects that qualify for the export and group them together based on their relationships.

Identifying Tables to Exclude From Migrations

Some maintenance objects that are eligible to be migrated may include child tables that should not be included in the migration. For example, if an object includes log tables, the entries in the log should reflect the actions on the object in that system, and will be different between the source system and the target system. If you have a custom Maintenance Object that includes tables you don't wish to migrate (such as a log table), use the **Non-Migrated Table** option on the MO to specify this table. All child records for this table will also be ignored during migration.

Another use case to consider is a child "many-to-many" table that connects two administrative objects and exists in the maintenance object of both tables. The child table may be in both MOs for convenience sake, but it may be that one MO is considered more of a "driver" object and the other more of a subordinate. If you are doing a migration where you want to copy a subset of objects, you may want to only copy the driver object and its children and their data but not their children. For example, a To Do Type includes a collection of valid To Do Roles and in turn the To Do Role refers to its To Do Types. If an implementation wants to copy a single To Do and include all its related information, including its To Do Roles, it does not want the migration of each To Do Role to in turn copy all its To Do Types (and their data).

Note: The MO option must be set in both the Source and Target systems for a given MO.

Configuring Custom Objects for Migration

During the implementation and extension of the product, new custom administrative maintenance objects may be introduced. If your implementation would like to migrate records in those maintenance objects using the Content Migration Assistant, additional steps must be performed, which are highlighted in the following sections.

Physical Business Object

As described in Understanding the BO Filtering Process, the migration plan requires a business object for its instruction. The business object is used to identify the records eligible for inclusion in the migration. Assuming your custom tables use one or more "logical" business objects for their processing, your implementation must decide if these business objects are appropriate for use by the migration plans, or if a physical BO is warranted. If so, create an appropriate physical BO.

Review MO Option Configuration

The following points highlight maintenance object (MO) configuration that should be reviewed or updated to support CMA:

- If a physical BO was created (above), link it to the MO as an option using the appropriate option type.
- Be sure that your MO defines an appropriate FK Reference and includes an Option on the MO that identifies the FK Reference. This is used by various portals and zones for CMA when showing detail about records being imported into the target region. Also be sure that this FK reference defines an Information program.
- As described in Identifying Tables to Exclude From Migrations, an MO option is used to identify child tables for an MO that should never be included in a migration. If your custom maintenance object includes a standard Log table, than the recommendation is to list that table as an excluded table. Depending on the specific design of the maintenance object, there may be other child tables to define.

Copyright © 2024, Oracle and/or its affiliates.

Characteristic Type Configuration

The CMA import process will attempt to create a log record for any administrative object that includes a log table. If your implementation has introduced any custom administrative tables that you plan to include in a migration request and it includes a log table, you must, to ensure that the log creation is successful, add your log table as a valid characteristic entity to the characteristic type **F1-MGO** (Migration Object).

Navigate to Characteristic Type and select the characteristic type **F1-MGO**. Navigate to the **Characteristic Entity** tab and add a row to include the characteristic entity for your custom maintenance object's log table.

Standard CMA Configuration

Create one or more migration plans for the new object, depending on the type of data in the maintenance object and the types of migrations you envision:

- If you have implemented only one "logical" business object used to define the data in the MO, then a single migration plan that references the this BO (or the maybe the MO's physical BO) is appropriate.
- If you have implemented more than one "logical" business object, would the data for multiple business objects get copied together? Then perhaps a single migration plan that references the MO's physical BO is appropriate.
- Are there additional foreign keys defined using mapXML in the business object(s) for the MO? If so, then it is recommended to include sub-instructions to define the links. At this point, if multiple "logical" BOs exist, your implementation may choose to define all the additional elements in the same migration plan or choose to define separate migration plans for each logical BO.
- Your implementation may decide to define more than one migration plan for the same type of record based on the types of migrations you plan to include. For example, you may decide to include a migration plan that copies only the records in this maintenance object. You may decide to define another migration plan that copies the records in this MO along with related records in another MO (for a special type of migration). Having said that, be sure to design the migration plan with reuse in mind to minimize maintenance efforts.

In order to support **Entity List** migration request, a default migration plan must be defined as an option on the maintenance object. This should be a single migration plan that supports all types of business objects for the MO.

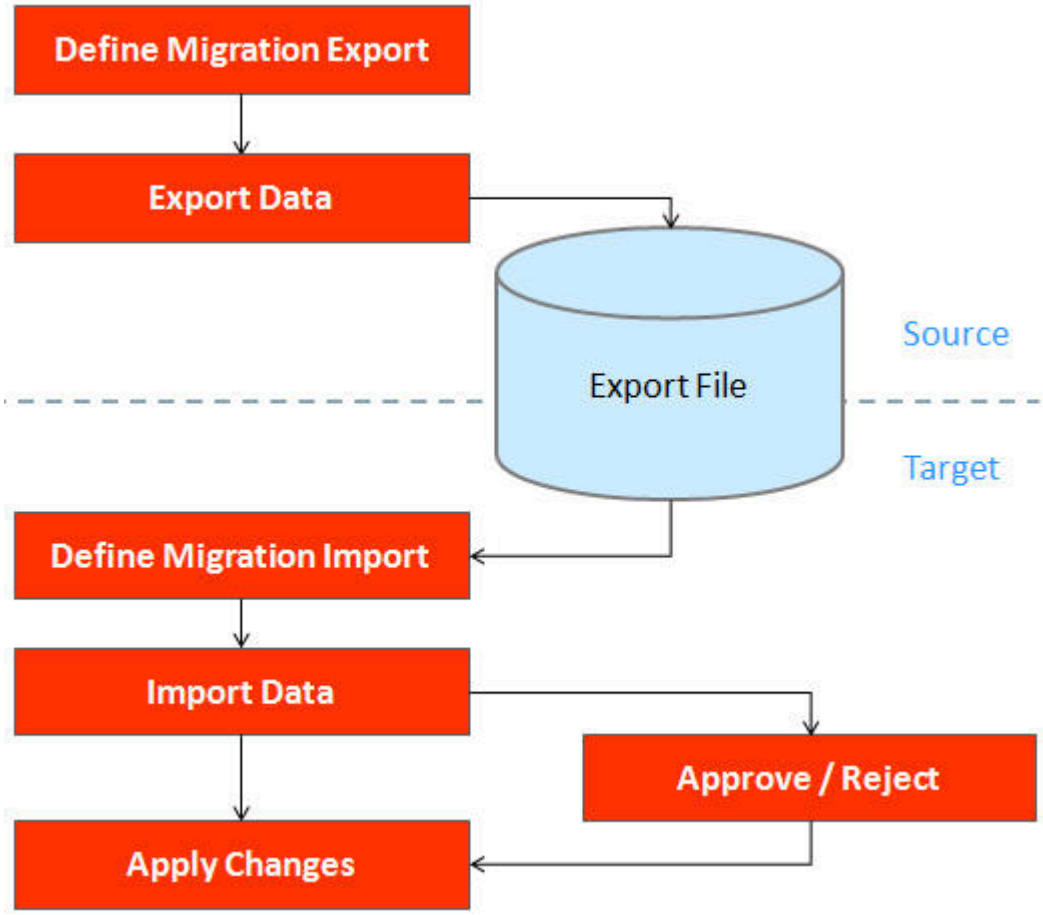
If your implementation has a template migration request to use for migrations, include the new migration plan(s) as appropriate.

CAUTION: Important! New migration plans and migration requests should follow naming conventions. Refer to System Data Naming Convention for more information.

The CMA Execution Process

The following diagram illustrates a high-level view of the Content Migration Assistant execution process. The subprocesses illustrated here are described in more detail in the following sections.

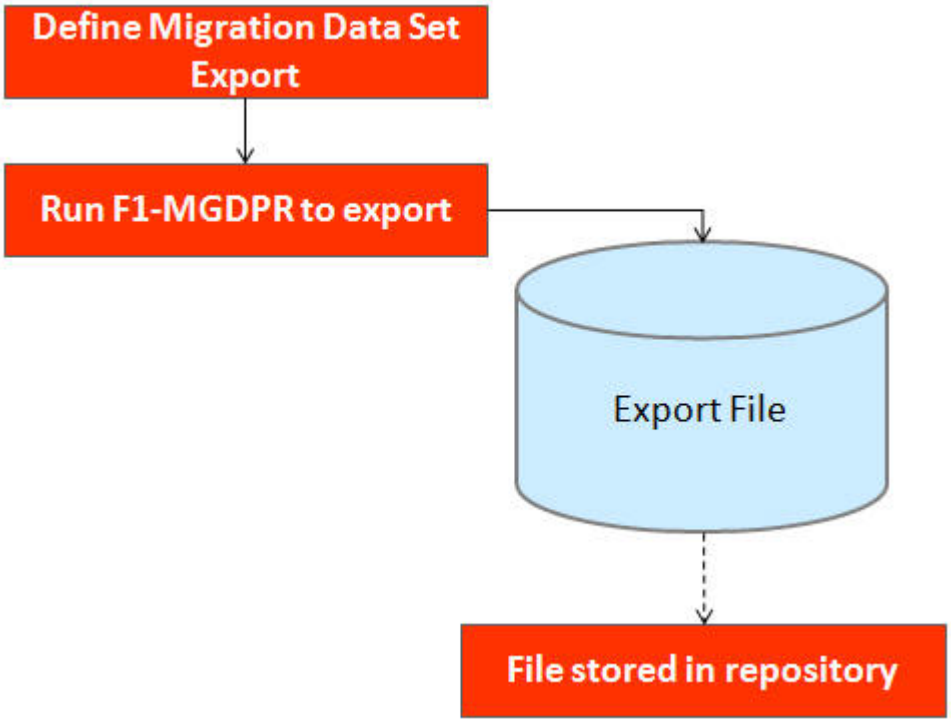
Copyright © 2024, Oracle and/or its affiliates.



Exporting a Migration

The migration export process begins in the source environment by defining a **Migration Data Set Export**, which specifies a defined **Migration Request** and provides a file name and description for the export file. After the data set is defined and saved, the **Migration Data Set Export Monitor** batch job can be submitted generate the export file. The lifecycle of the Migration Data Set Export business object includes algorithms that select the appropriate records according to the migration request, determine dependencies between records to build groupings of related objects and create the export file.

Copyright © 2024, Oracle and/or its affiliates.



The following topics provide more detail about this process.

Migration Data Set Export

To migrate data from one region to another, define a **Migration Data Set Export**. This establishes the export file name and identifies the migration request.

To view an existing migration data set export, navigate using **Admin** > **Implementation Tools** > **Migration Data Set Export**. Use the query criteria to locate the desired data set.

Note that you can also initiate the creation of an export data set from the Migration Request portal using the **Export** button.

The export requires the name of an existing **Migration Request**.

Enter a unique **File Name** for the export. Do not use spaces in the name, and do not enter the file extension or a path. The output location and file extension of the intended export file, which should appear in the **Export Directory** and **File Suffix** labels, are defined as described in the topic Migration Assistant Configuration.

Enter an **Export Description** to provide information about the purpose of the export. Note that this field is not language enabled.

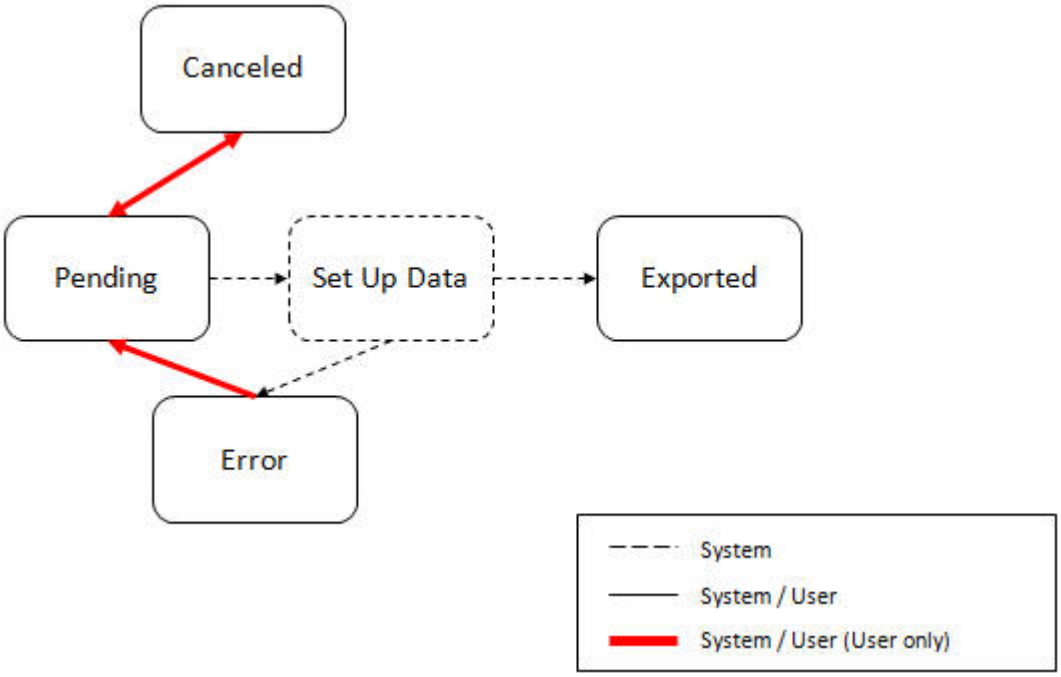
The **Source Environment Reference** is for information purposes. It should be populated with text that provides a meaningful description of the source environment. The default value is the URL of the source environment.

The **Migration File Version** is only visible if the product currently supports creating a file in an earlier file format version. If this element is visible and you are planning to import the file in a target environment that uses an older migration file format version, indicate the lower file format version that you wish the export to use.

Export Lifecycle

The following diagram describes the lifecycle of a Migration Data Set Export (data set).

Copyright © 2024, Oracle and/or its affiliates.



The following points describe the lifecycle.

- The data set is created in **Pending** status.
- A user may choose to temporarily **Cancel** a pending data set to prevent it from being processed. The user can later return it to the Pending state when desired.
- The record remains in Pending until its monitor batch job is submitted. The **Migration Data Set Export Monitor** (F1–MGDPR) selects pending records and transitions them to **Set up data**. Refer to Running Batch Jobs for more information.
- Set up data is a transitory state that includes the algorithm that does the work of determining the objects to include in the export and group related objects together into a transaction. If everything is successful, the export file is written to the appropriate file location and the record transitions to **Exported**. If an error is detected, the process stops and the record transitions to **Error**.
- If a record is in error and it is possible to correct the error, the record may be transitioned back to Pending to try again.

When the process is marked as **Exported**, the export file can be imported into the target system.

Note: The export process creates a file, providing the benefits of having a standalone file. It can be stored in a version control system for audit purposes or provided to others for import purposes.

CAUTION: Under no circumstances should exported data files be edited manually. Doing so could cause data corruption when the file content is applied to the target environment.

Note: The export functionality is supported using the business object **Migration Data Set Export** (F1-MigrDataSetExport). The expectation is that implementations will use the delivered base business object and its logic and will have no reason to implement a custom business object for the CMA export process.

Importing and Applying a Migration

The import process is broken down into four general steps: Import, Compare, Approve, Apply. The following points provide an overview of the steps.

- **Import**. The first step covers importing the file and creating appropriate Migration Import records in the target environment to facilitate the subsequent steps.

Copyright © 2024, Oracle and/or its affiliates.

- **Compare**. The compare step reviews each object that is in the import file and compares the object in the import against the equivalent record in the target environment. The comparison step results in noting which objects are unchanged, which are new (and the appropriate SQL to insert them) and which objects are changed (and the appropriate SQL to update them). Based on user configuration at import time, the objects that qualify for the import may be in a state that requires review or may be pre-approved.
- **Approve**. Once the comparison is complete, the user should review the results. There may be records marked for review. All of these records must be approved or rejected before the import can proceed. Users may choose to suppress individual SQL statements for a given object that is approved. When the user is satisfied with the results of the comparison and has completed the review, the import is marked to proceed to the Apply step. Optionally, a migration import may be configured to automatically apply.
- **Apply**. This is the final step and is the step where the records in the target environment are added or updated. Because of potential high volumes of data and because of possible dependencies between records, this step supports two levels of attempting to apply the records. There is an apply step at the object level and an apply step at the transaction level. This will be described in more detail below.

Import Step

The import process starts with verifying the import directory configured as described in the following topic Master Configuration - Migration Assistant and ensuring that the exported file is located in that directory. Then, in the target environment, a **Migration Data Set Import** record should be created. The user indicates the file name.

In addition, the user decides what the default status should be for resulting objects.

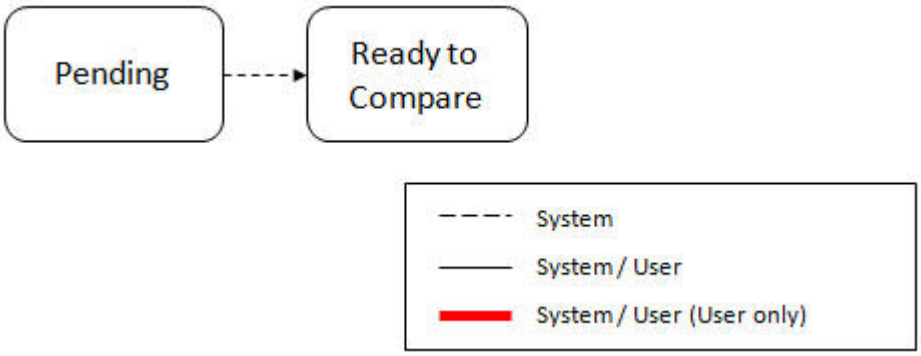
- The **Default Status for Add** sets the default status for objects that are determined to be ***new*** during the import comparison process. The default is to automatically set new objects to **Approved** status. Other options are to set any new objects to **Rejected** or **Needs Review** status.
- The **Default Status for Change** sets the default status for objects that are determined to be ***changed*** during the import comparison process. As with new objects, the default for changed objects is **Approved**, with **Rejected** or **Needs Review**options available.

The user may also configure the **Automatically Apply** flag to **Yes**. This allows for use cases where the migration is repetitive and has been tested and the user feels that there is no need for manual approval. Note that when configuring this setting, neither of the Default Status values may be set to **Needs Review** and at least one must be set to **Approved**.

The file to import contains a list of all the objects included in the export. Any objects that the export step determined to be related have been grouped into “transactions”. Once the Migration Data Set Import is created, the next step is for the system to read in the file and create Migration Transactions and Migration Objects.

By default, the import step creates a Migration Object for each entity in the file. When importing a high volume data set of business entities, the user may request to import entities in bulk mode for performance reasons. In this mode, a single Migration Object record may contain multiple entities from the file. If at least one of the entities included in the migration object record is new and the remaining ones are either new or unchanged then the **Default Status for Add** setting applies to it. If at least one included entity is determined to be changed then the **Default Status for Change** applies to the record. Refer to bulk import mode for more information.

The following is a portion of the Migration Data Set Import lifecycle as it pertains to the import step.



Copyright © 2024, Oracle and/or its affiliates.

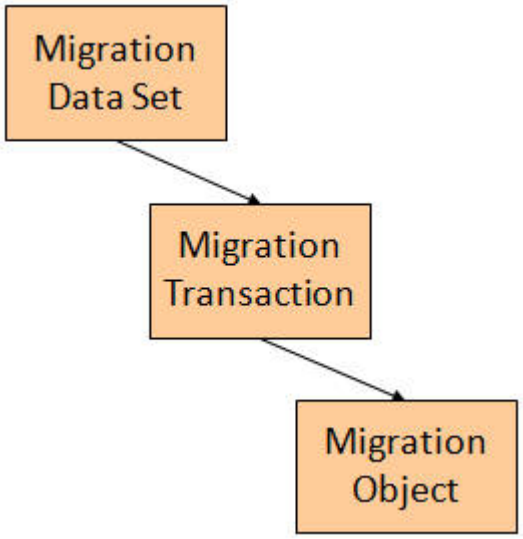
The following points describe the lifecycle.

- The data set is created in **Pending** status.
- The record remains in Pending until its monitor batch job is submitted. The **Migration Data Set Import Monitor** (F1–MGDIM) selects pending records and transitions them to **Ready to Compare**. Refer to Import Process Summary and Running Batch Jobs for more information.

  The Ready to Compare state has an algorithm that is responsible for reading the related import file and creating the migration transactions and migration objects. The data set remains in this state until the comparison step is complete.

Note: A user may choose to **Cancel** a data set. Refer to Cancelling a Data Set for more information.

The following diagram highlights the relationships of the resulting migration import records.



The migration transaction and migration object each have their own lifecycle that will help manage the subsequent compare and apply steps. At the end of the import step, the status values of the three types of records are as follows:

- Migration Data Set Import is in the **Ready to Compare** state.
- Migration Transaction is in the **Pending** state.
- Migration Object is in the **Pending** state.

Note: The import functionality is supported using business objects supplied by the base product. The expectation is that implementations will use the delivered base business objects and their logic and will have no reason to implement a custom business objects for the CMA import process. The base business objects are **Migration Data Set Import** (F1-MigrObjectImport), **Migration Transaction** (F1-MigrTransactionImport) and **Migration Object** (F1-MigrObjectImport).

Compare Step

The import step results in the creation of one or more migration objects. Typically, one migration record is created for each record selected in the export based on the export's migration request and its configuration. Related objects are grouped together in migration transactions.

When importing a high volume data set of business entities, the user may request to import entities in bulk mode for performance reasons. In this mode, a single Migration Object record may contain multiple entities from the file. Refer to bulk import mode for more information.

The next step in the import process is the Comparison step. In this step, the data captured by the import file for each object is compared to the view of that object in the target environment.

To cater for a possible large volume of objects, the comparison is done via a batch monitor. To aide in performance of the process, the monitor is performed on the migration objects so that it can be run multi-threaded. Once the objects are

Copyright © 2024, Oracle and/or its affiliates.

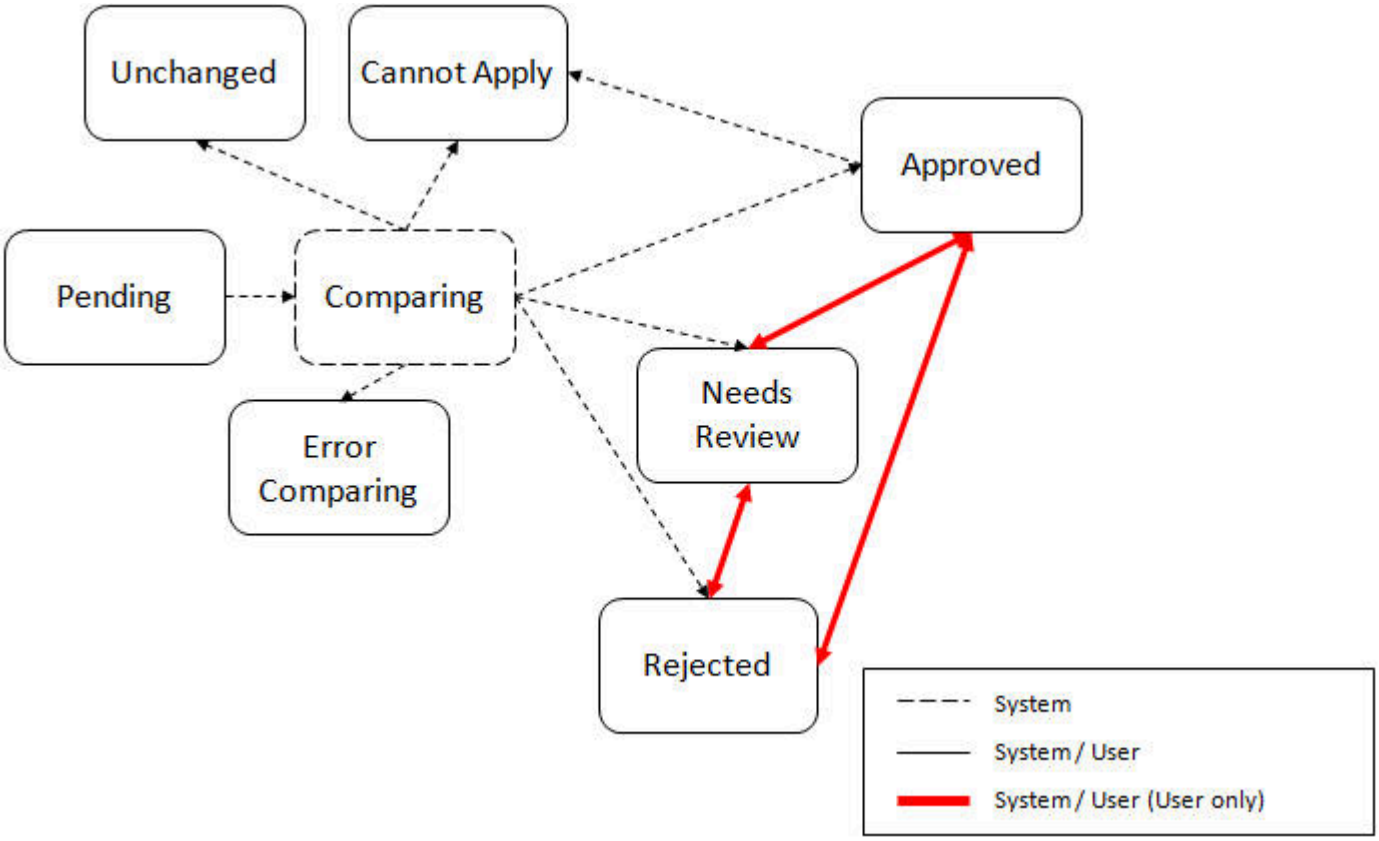
finished with the comparison, the migration transactions and the migration data set should be updated with an appropriate overall status before continuing to the next step. As a result, the comparison actually requires three steps: Migration Object Comparison, Migration Transaction Status Update and Migration Data Set Export Status Update. The steps are explained in detail in the following sections.

Note: Refer to Running Batch Jobs for more information about streamlining the various steps in the process.

Migration Object Compare

This is the main step of the comparison. The **Migration Object Monitor** (F1–MGOPR) selects pending migration object records and transitions them to **Comparing**. This is a transitory state that includes the algorithm that does the work of comparing. There are various possible outcomes that could occur based on the logic in the algorithm. The following diagram illustrates a portion of the migration object lifecycle that pertains to comparison.

Note: When using separate batch processes for business data the **Migration Object Monitor (Business)** process (F1-MGOPB) works in the same way to compare business migration objects.



The following points describe the lifecycle.

- When **Pending** records are selected by the monitor batch job, it transitions to **Comparing**. If the migration object refers to one or more pre-compare algorithms, they are executed to adjust the data prior to comparison. Then algorithm will determine the appropriate next state by comparing the source data to the target data.
- Entity related notes:
  - If the record in the migration object is found in the target environment and the data is exactly the same, the record transitions to **Unchanged** (with the object action value also set to **Unchanged**).
  - If the record in the migration object is found in the target environment and the data is different, the algorithm sets the object action value to **Change** and generates the appropriate SQL to be used later in the Apply step to update the record. It then transitions to **Approved**, **Needs Review** or **Rejected** based on the Default Status For Change setting captured on the Data Set.

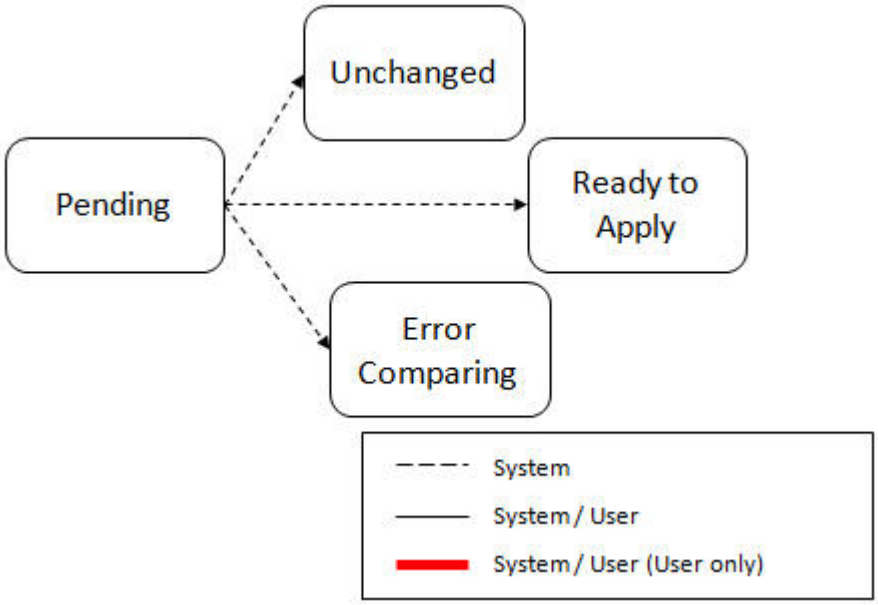
Copyright © 2024, Oracle and/or its affiliates.

- If the record in the migration object is not found in the target environment, the algorithm sets the object action value to **Add** and generates the appropriate SQL to be used later in the Apply step to insert the record. It then transitions to **Approved**, **Needs Review** or **Rejected** based on the Default Status For Add setting captured on the Data Set.
- When the import is performed in bulk mode each included entity is compared separately but the overall status of the record is set as follows:
  - If all included entities are unchanged then the migration object record transitions to **Unchanged** (with the object action value also set to **Unchanged**).
  - If all included entities are either unchanged or changed then the object action value is set to **Change**), corresponding SQLs are generated for the changed entities only and the migration object transitions based on the Default Status For Change setting captured on the Data Set.
  - If all included entities are either unchanged or new then the object action value is set to **Add**), corresponding SQLs are generated for the new entities only and the migration object transitions based on the Default Status For Add setting captured on the Data Set.
  - If some of the included entities are new and some changed then the object action value is set to **Mixed Actions**), corresponding SQLs are generated for the each entity and the migration object transitions based on the Default Status For Change setting captured on the Data Set.
- If there is any issue with attempting to parse the object data from the import, the record transitions to **Error Comparing**.
- If there is any reason that the imported object is not valid for import, the record transitions to **Cannot Apply**. The log will be updated with the error that caused the record to transition to this state. An example is that perhaps the record was exported in a different version of the product and has additional elements that are not recognized in this version.

Note: Refer to Cancelling a Data Set for information about cancelling a data set and its impact on its related objects.

Migration Transaction Status Update

After the import step, the migration transaction remains in the Pending state until all its objects have completed the comparison step. At that point, the status of the transactions should be updated based on the results of their objects. The **Migration Transaction Monitor** (F1–MGTPR) selects pending migration transaction records and runs its monitor algorithms. There are various possible outcomes that could occur based on the logic in the algorithms. The following diagram illustrates a portion of the migration transaction lifecycle that pertains to comparison.



The following points describe the lifecycle possible next states after Pending.

- If any related migration object is in the Error Comparing state, the transaction transitions to **Error Comparing**.

Copyright © 2024, Oracle and/or its affiliates.

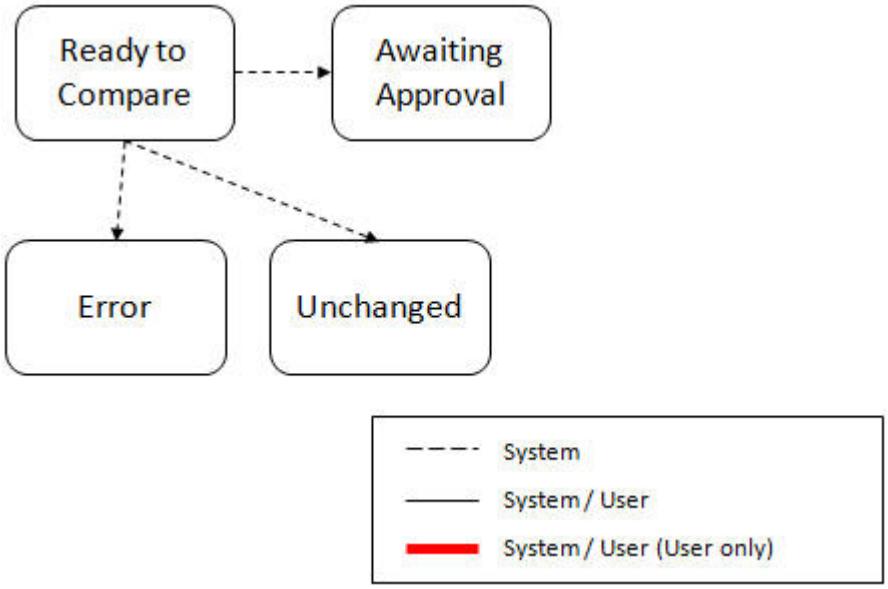
- If all related migration objects are in the Unchanged state, the transaction transitions to **Unchanged**.
- Otherwise, the transaction transitions to **Ready to Apply**. This means that at least one object is in an "apply-able" state.

The transaction remains in the **Ready to Apply** state until a user has approved the data set to move to the Apply step and the transaction's related objects have attempted to apply themselves. This is described in more detail below.

Note: Refer to Cancelling a Data Set for information about cancelling a data set and its impact on its related objects.

Migration Data Set Import Status Update

Once all the objects and all transactions have been updated via the previous two steps, the migration data set export must be updated based on the results of their transactions. The **Migration Data Set Import Monitor** (F1-MGDIM) selects Ready to Compare data sets and runs its monitor algorithms. Note that this is the same monitor process that is used to select Pending data sets. There are various possible outcomes that could occur based on the logic in the algorithms. The following diagram illustrates the portion of the migration transaction lifecycle that pertains to comparison.



The following points describe the lifecycle possible next states after Ready to Compare.

- If any related migration transactions is in the Error Comparing state, the data set transitions to **Error**.
- If all related migration transactions are in the Unchanged state, the data set transitions to **Unchanged**.
- Otherwise, the transaction transitions to **Awaiting Approval**. This means that there are no errors and at least one object is in an "apply-able" state.

The data set remains in the **Awaiting Approval** state until a user decides that the data set and all its records are ready to progress to the Apply step.

Note: A user can choose to cancel a data set at any time while it is in progress. Refer to Cancelling a Data Set for more information.

Approval Step

Once the comparison is complete and the data set transitions to the Awaiting Approval state, a user needs to progress the data set to **Apply Objects** to trigger the Apply step. The following points describe steps a user may take during the approval step.

- If the data set configuration for the Default State for Add and Change was set to **Approved**, then any migration object that is determined to be eligible for the Apply step will be in the Approved state. In this situation, a user may

Copyright © 2024, Oracle and/or its affiliates.

want to review the data set and its transactions and objects to see verify that the results make sense. At that time, the user is able to move an object to Needs Review or Rejected as appropriate.

- If the data set configuration for the Default State for Add and Change was set to **Needs Review** for either option, then each migration object in the Needs Review state must be reviewed and the user must either Reject or Approve each object before moving to the Apply step.
- If the data set configuration for the Default State for Add and Change was set to **Rejected** for either option, the assumption is that the rejected records don't need to be reviewed. But if a user finds a rejected record that shouldn't be rejected, it may be transitioned to Approved (or Needs Review) as appropriate.

A user has the option to mark one or more SQLs within a migration object to be suppressed. In this case, if the migration object is approved, then only the SQLs that are not marked as Suppressed will be applied. This option is useful when importing a record that has many child records. It may be that the user wants to only include a subset of the child records. This may be done by suppressing the child records that should not be included.

Once the user is comfortable with the data set's results and no more objects are in the Needs Review state, the user should transition the record to **Apply Objects**. This will initiate the Apply step.

Alternatively, if the Automatically Apply flag was set to **Yes** when creating the import record, the import data set will progress from **Awaiting Approval** to **Apply Objects**. Refer to Import Process Summary for more information.

Note: Refer to Maintaining Import Data for details about the pages provided to help the user review a data set and its transactions and objects to help in the approval step.

Note: A user can choose to cancel a data set at any time while it is in progress.

Apply Step

The apply step is the step where records in the target environment are added or updated. Like the comparison step, the apply step is actually multiple steps to optimally handle high volume and dependencies between records as smoothly as possible.

Note: Refer to Running Batch Jobs for more information about streamlining the various steps in the process.

Before explaining the apply steps in detail, the following points highlight the type of data that may be included in a given data set.

1. Records that have no foreign keys and therefore no dependencies on other records. Examples: Message, Display Profile.
2. Records that have foreign keys that may already be in the target. Examples: Algorithms for existing algorithm types, To Do Roles for existing To Do Types.
3. Records that have foreign keys that are new records but also part of the migration. CMA detected the relationship at export time and grouped the objects in the same transaction. Example: Script-based Algorithm Type where the script is also in the migration.
4. Records that have foreign keys that are new records but also part of the migration. CMA did not detect the relationship. This may occur if the referenced foreign key is in a XML or parameter column and the migration plan did not include an instruction to explicitly define the relationship. Example, a Zone that references a visibility script.
5. Records that have circular references where both records are new and are part of the migration. CMA detected the relationship at export time and grouped the objects in the same transaction. Example: plug-in Script for a BO plug-in spot. The script references the BO and the BO references an algorithm for the script's algorithm type. Another example is when the same record is maintained by multiple maintenance objects and therefore exists in multiple migration objects.

To handle high volume data, the first step in the apply process is to perform the apply logic at the migration object level via a multi-threaded batch job. This should result in all records in categories 1 and 2 above being applied successfully.

For records in categories 3 and 4 above, if a record with a foreign key is added or updated before its related record, the validation will fail. However, if the related record is added first and then the record referring to it is added, validation will pass. The tool handles these dependencies as follows:

Copyright © 2024, Oracle and/or its affiliates.

- Dependency between master and transaction entities is typically hierarchical and in most cases straight-forward. The tool leverages that knowledge to orchestrate the processing of objects in an optimal way that follows their dependency order as much as possible. Note that relationship between entities could be complex and this approach does not eliminate all processing order related errors but rather significantly reduce them.
- Dependency between configuration entities is more complex and inter-twined and therefore migration objects are not ordered, i.e. the multi-threaded batch process may not process records in the desired order.
- To overcome the potential issue of processing order related errors, the Apply step has special functionality, described in detail below.

For records in category 5 above, the circular reference will mean that the apply process at the object level will not successfully add or update these records. The apply process at the transaction level will cover these records. This is described in detail below.

Apply Objects

Once the Data Set is in the state of **Apply Objects**, the **Migration Object Monitor - Apply** process (F1-MGOAP) runs to attempt to apply the objects.

Note:

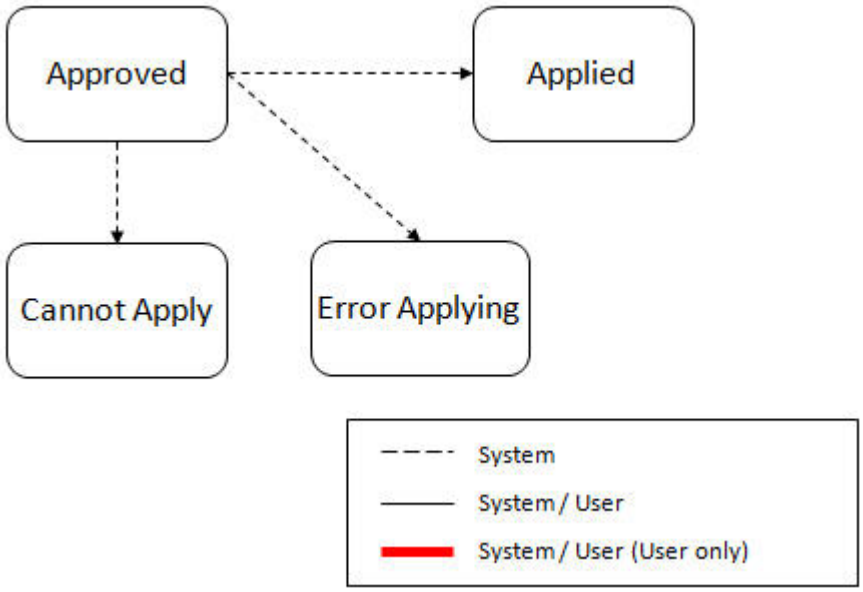
When using separate batch processes for business data the **Migration Object Monitor (Business)- Apply** process (F1-MGOAB) works in the same way to apply business migration objects.

The background process in conjunction with the Apply algorithm have special functionality to ensure records in categories 3 and 4 (above) successfully apply during this step:

- The **Migration Object Monitor - Apply** process is a special one that continually re-selects records in the **Approved** state until there are no more eligible records to process.
- When an error is received in the Apply Object algorithm, the algorithm increments an "iteration count" on the migration object record. If the iteration count does not exceed a maximum count (noted in the algorithm), the object remains in the **Approved** state and is eligible to be picked up for processing again. If the iteration count exceeds the maximum defined in the algorithm, the record transitions to the **Error Applying** state.

Note: When submitting this Apply batch job, be sure to set the number of threads to a number that does not exceed the number of threads supported by the thread pool worker. Doing this will cause the ‘excess’ threads to wait for the supported number of threads to finish.

The following diagram is the portion of the migration object lifecycle that pertains to the Apply step.



Copyright © 2024, Oracle and/or its affiliates.

At the completion of the Apply monitor process, typically the objects will be in the **Applied** state or the **Error Applying** state. The records in the Error Applying state are in that state for one of two reasons.

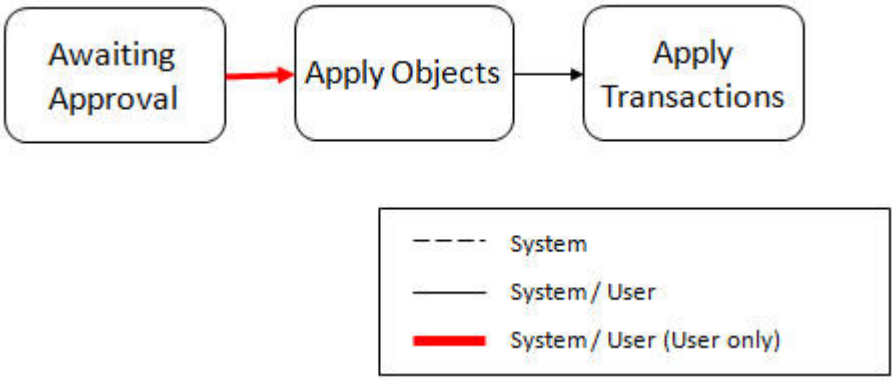
- They are in category 5 described above where the records have a circular reference with another record. For this scenario, the Apply Transactions step described below should successfully apply the records.
- There is some other error that is unrelated to the records in the current migration. In this case, manual intervention may be required. Refer to the Resolving Errors section below for more information.

As shown in the diagram, the Apply Objects algorithm may also detect a reason that the object cannot be applied. This may occur if the object in the target environment has been updated since the comparison step, making the SQL captured at that point no longer applicable. If this occurs, after the current migration is fully applied, the original file may imported again, and new comparisons can be generated and applied.

Apply Transactions

Ideally, after the Apply Objects step, all the objects are **Applied** or are in **Error Applying** due to the "circular reference" situation. The typical next step is to turn over responsibility to the transactions. The migration transactions can then attempt to apply their objects in bulk.

In order to ensure that multiple background processes are not trying to select migration objects to run the Apply step, the Transactions are only eligible to attempt to "apply my objects" if the Data Set is in the **Apply Transactions** state.



A monitor algorithm (executed by the data set monitor batch process) on the Apply Objects state checks to see if all migration objects are no longer **Approved** or the count of records in **Error Applying** does not exceed a configured limit. If so, it automatically transitions the record to the **Apply Transactions** state.

If the number of objects in **Error Applying** exceeds a configured limit, the monitor algorithm does not automatically transition the record. In that case, a user must determine if the large number of errors can be resolved or manually transition to **Apply Transactions** (despite the large number of errors). The Resolving Errors section below describes alternative steps that the user may take if there are errors.

Once the Data Set is in the state of **Apply Transactions**, the **Migration Transaction Monitor - Apply** process (F1-MGTAP) runs. It attempts to apply the transaction's objects. If no migration objects are in error, the migration transaction simply transitions to **Applied**. If any of the migration objects are in **Error Applying**, the background process and the Apply algorithm have special functionality to try to overcome dependencies in migrated objects:

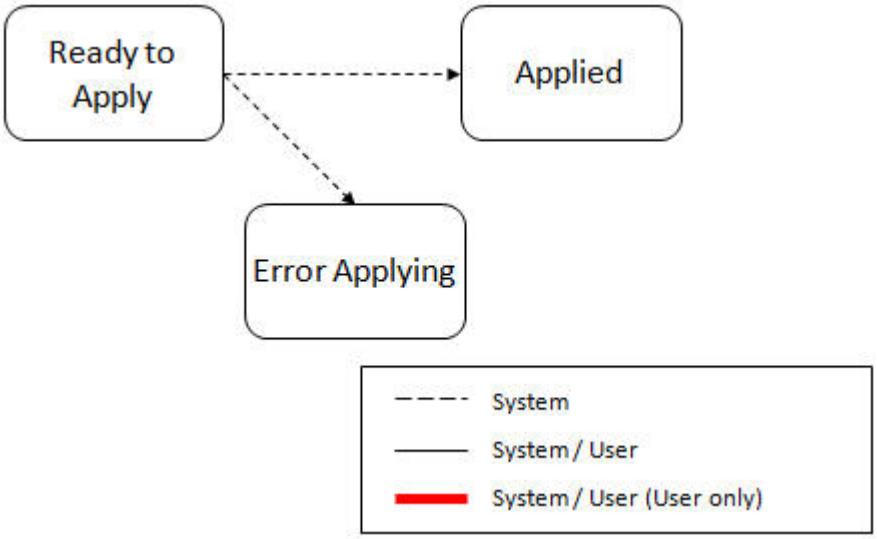
- The Apply algorithm selects all migration objects in error and performs all their SQL, then validates all the records. If there are objects in the transaction with circular references, they should pass validation at this point.
- Because there may still be some dependencies across transactions, similar error handling described in the Apply Objects step occurs here. When an error is received in the Apply Transaction's Object algorithm for any of the objects in the transaction, the algorithm increments an "iteration count" on the migration transaction record. If the iteration count does not exceed a maximum count (noted in the algorithm), the transaction remains in the **Ready to Apply** state and is eligible to be picked up for processing again. If the iteration count exceeds the maximum, the record transitions to the **Error Applying** state. Note that if any objects in the transaction are in error, none of the objects are applied. They all remain in error.

Copyright © 2024, Oracle and/or its affiliates.

- The **Migration Transaction Monitor - Apply** process is a special one that continually re-selects records in the **Ready to Apply** state until there are no more eligible records to process.

> **Note:** When submitting this Apply batch job, be sure to set the number of threads to a number that does not exceed the number of threads supported by the thread pool worker. Doing this will cause the 'excess' threads to wait for the supported number of threads to finish, erasing the benefit of the iteration processing.

The following diagram is the portion of the migration transaction lifecycle that pertains to the Apply step illustrating the points above.



If at the end of the transaction level Apply process there are transactions in error (and therefore there are still objects in error), a user must review the errors and determine how to fix them. Refer to the Resolving Errors section below for more information.

Resolving Errors

As mentioned in the previous sections, errors may be received after the Apply Objects process runs. If the number of records in error is below a certain limit (and the data set monitor batch job is submitted to execute the monitor algorithms) the system will automatically transition the data set to the **Apply Transactions**. If the monitor batch job is not run or if the number of objects in error exceeds a certain limit, a user must make the decision after viewing the errors in the Objects in Error zone on the Migration Data Set Import portal.

- If the errors appear to be dependency related, the user can decide to let the "transactions apply their objects" and transition the data set to **Apply Transactions**, described above.
- If the errors appear to be related to an outside issue that can be manually resolved, the user may choose to fix the issue and redo the Apply Objects step.
- The user may also decide to reject one or more objects to remove them from the migration.

After the Apply Transactions step, if there are still errors, a user must review the records and determine how to proceed. Errors are visible in the **Transactions in Error** zone on the Migration Data Set Import portal.

- The user may decide to reject one or more objects to remove them from the migration.
- The user may manually resolve an issue external to the migration and then decide to do one of the following:
  - Redo the **Apply Objects** step. This is recommended if there are a large number of Objects still in error and not a large number of dependencies expected. The benefits of running the Apply Objects multi-threaded will ensure that the process runs efficiently.
  - Redo the **Apply Transactions** step.

Because the objects and transactions are in Error Applying, in order to "retry" the Apply step after manually fixing an error, the system needs to move the records back to the state that allows them to be picked up by the appropriate Apply

Copyright © 2024, Oracle and/or its affiliates.

process. For migration objects, records need to be moved back to **Approved**. For migration transactions, records need to be moved back to **Ready to Apply**. The following points describe the Retry logic for migration objects.

- If a user decides to **Retry Objects** (using an action button on the Migration Data Set Import page), the data set transitions to the **Retry Objects** state. At this point the Migration Object monitor must be run.
- The monitor on the **Error Applying** state for the objects detects that the data set is in the state of **Retry Objects** and that triggers the transition back to **Approved**.
- The next step is to transition the data set from **Retry Objects** to **Apply Objects**. This may be done manually or by running the Migration Data Set Import monitor process.
- Now the objects are eligible to be picked up by the object level Apply process.

Analogous logic exists for the migration transactions.

- If a user decides to **Retry Transactions** (using an action button on the Migration Data Set Import page), the data set transitions to the **Retry Transactions** state. At this point the Migration Transaction monitor must be run.
- The monitor on the **Error Applying** state for the transactions detects that the data set is in the state of **Retry Transactions** and that triggers the transition back to **Ready to Apply**.
- The next step is to transition the data set from **Retry Transactions** to **Apply Transactions**. This may be done manually or by running the Migration Data Set Import monitor process.
- Now the transactions are eligible to be picked up by the transaction level Apply process.

The retry logic may also occur when transitioning between the Apply Objects and Apply Transactions depending on whether or not there are errors. The following scenario highlights this point.

- After the **Apply Objects**step there are objects in **Error Applying**. The data set transitions to **Apply Transactions** and the Apply step is done at the transaction level.
- After the **Apply Transactions** step there are transactions in **Error Applying**.
- User chooses to try to apply objects again (by clicking **Retry Objects**). The steps outlined above for retrying objects are followed at this point.
- After the apply objects, user may choose to retry objects again (after fixing errors if applicable).
- At some point the user will transition to**Apply Transactions** again. If there are transactions in **Error Applying**, the system will automatically transition the data set to **Retry Transactions** and the steps outlined above for retry transactions are followed.

Finalize Apply Step

Once all the migration objects for a migration transaction are in a final state (**Applied**, **Rejected** or **Cannot Apply**), the migration transaction transitions to the **Applied** state. Once all the migration transactions are in the **Applied** state, the Migration Data Set record transitions to the **Completed** and the import is complete.

Note: To review the full lifecycle for each record, refer to the Business Object - Summary tab in the application metadata for the base business objects **Migration Data Set Import** (F1-MigrObjectImport), **Migration Transaction** (F1-MigrTransactionImport) and **Migration Object** (F1-MigrObjectImport).

Adjusting Data Prior to Comparing

There are reasons that the data you are importing requires some adjustment before applying it in the target environment. In such cases, an algorithm plugged into the migration plan primary instruction may be used to adjust the data when importing. This algorithm is executed by the comparison algorithm before any comparison is performed. Algorithms of this system event receive the view of the source record (being imported) and the view of the existing record in the target region, if it exists. The data is provided using the physical BO of the migration plan's maintenance object. The algorithm may make changes and pass a new view of the record that should be used for the comparison. This system event supports multiple algorithms that are executed in sequence. Each algorithm receives the original record's data, the target record's data (if applicable) and the 'new' view of the data (as populated by previous algorithms, if any). The final 'new' view of the data is used for the object comparison. The system includes the algorithms linked to the primary instruction of the migration plan from the source environment. If the system finds any additional pre-compare algorithms linked to the migration plan's primary instruction in the target system, those algorithms are also executed.

Copyright © 2024, Oracle and/or its affiliates.

Fastpath: Refer to Base Business Objects for more information about physical BOs.

Some examples of records that may require import algorithms.

- Batch Control references its next batch sequence number along with snapshot information like the last run date / time. This information is only relevant with respect to its environment. The instruction for a batch control can include an algorithm to not overwrite the batch sequence number when copying a batch control.
- Some products include administrative objects that reference a master data object. Master data objects are not copied as part of CMA. An import algorithm may be used to adjust the referenced master data foreign key when importing, for example to reset it (or not overwrite when updating). If the algorithm knows how to find the appropriate master data record to link, that may also be included.
- If you are importing data that from a source environment on a lower version, if the object you are importing has been enhanced in the latest release with a new required field, you may need to provide an algorithm to populate data in that field.

Note that it is possible to use the algorithm to "reset" the source data as a way of indicating that the record should not be imported. For these situations, the migration object comparison step will transition the record to **Unchanged** and will use an object action value of **Canceled**. (Note that object action is a simple lookup value. The record is not transitioned to the **Canceled** BO state as to reserve that status for user initiated cancellations of the object or one of its parent records). This technique not expected to be used often because ideally using appropriate selection criteria at export time should ensure that the only records exported are those that should be imported.

Note: Legacy 'Import' system event. The system originally provided an Import system event / plug-in spot. The purpose of algorithms for this plug-in spot were similar in that they were meant to adjust imported data prior to adding or updating. The algorithms were executed in the Apply step. The logic does not allow for easily interacting with the record using a BO. This makes it difficult to use a plug-in script as the plug-in type. In addition, it is difficult to update elements in an XML column. The support for the plug-in spot will be removed in a future release. Algorithms to adjust the data should be using the pre-compare system event.

Import Process Summary

The following table summarizes the steps required to complete the import process from start to finish. Note that this section **only a summary** and assumes that you are familiar with the details described in the previous sections. It highlights what steps are manual and what steps are performed by a batch monitor process. For each step, the table highlights the Next Action sequence that would occur. For the Apply steps, there are two parts where multiple next actions are possible based on whether there are errors and the user's decision on how to resolve the error. Refer to Resolving Errors for more information. The possible next actions have the same sequence with a letter following the sequence highlighting the action to take based on the results of the previous step.

Note: When running the Apply batch jobs, be sure to set the number of threads to a number that does not exceed the number of threads supported by the thread pool worker.

Also note that a sequence and action marked in bold is considered the "normal path".

When using separate batch processes for business data any reference to a migration object monitor batch process applies also to the business data related batch controls.

Step	Seq	Action	Manual / Batch	Portal - Action	Batch Control	Record Impacted - Resulting Status	Next Action Sequence
Import	**10**	**Create Import record**	Manual	Migration Data Set Import - Add		Migration Data Set Import - Pending	11

Copyright © 2024, Oracle and/or its affiliates.

Step	Seq	Action	Manual / Batch	Portal - Action	Batch Control	Record Impacted - Resulting Status	Next Action Sequence
	11	**Import file**	Batch		F1-MGDIM - Migration Data Set Import Monitor	Migration Data Set Import - Ready to Compare Migration Transaction - Pending Migration Object - Pending	20
Compare	**20**	**Migration Object Compare**	Batch		F1-MGOPR - Migration Object Monitor [When using separate business data processes submit also F1-MGOPB (Migration Object Monitor (Business))]	Migration Object - Approved, Needs Review, Rejected, Unchanged or Error Comparing	21
	21	**Migration Transaction status update**	Batch		F1-MGTPR - Migration Transaction Monitor	Migration Transaction - Ready to Apply, Unchanged or Error Comparing	22
	22	**Migration Data Set Import status update**	Batch		F1-MGDIM - Migration Data Set Import Monitor	Migration Data Set Import - Awaiting Approval, Apply Objects (if configured for Automatic	30

Copyright © 2024, Oracle and/or its affiliates.

Step	Seq	Action	Manual / Batch	Portal - Action	Batch Control	Record Impacted - Resulting Status	Next Action Sequence
						Apply), Unchanged or Error	
Approval	**30**	**Review comparison results, approve / reject as needed** (This step is skipped if the data set is configured for Automatic Apply)	Manual	Migration Data Set Import, drill in to the Transactions Objects as necessary.		Migration Object - Approved or Rejected (no records should be in Needs Review) Migration Data Set Import - Apply Objects	40
Apply	**40**	**Apply Objects**	Batch		F1-MGOAP - Migration Object Monitor - Apply [When using separate business data processes submit also F1-MGOAB (Migration Object Monitor (Business) - Apply)]	Migration Object - Applied or Error Applying	41 Appropriate next action is based on error review, if applicable.
	41a	Migration Data Set Import status update - auto transition to Apply Transactions. Only applicable if the number of migration objects in Error	Batch		F1-MGDIM - Migration Data Set Import Monitor	Migration Data Set Import - Apply Transactions	42

Copyright © 2024, Oracle and/or its affiliates.

Step	Seq	Action	Manual / Batch	Portal - Action	Batch Control	Record Impacted - Resulting Status	Next Action Sequence
		Applying is below a threshold					
	41b	**Migration Data Set Import status update - manual transition to Apply Transactions.** Occurs if user reviews errors and determines that they may be resolved in the Apply Transaction step.	Manual	Migration Data Set Import - click Apply Transactions			
		System detects that all the transactions are in the Ready to Apply state and proceeds to the Apply Transactions state.				Migration Data Set Import - Apply Transactions	42
		System detects that there are transactions in the Error Applying state and transitions instead to the Retry Transactions state.				Migration Data Set Import - Retry Transactions	45
	41c	Migration Data Set Import status update - manual transition to Retry Objects. Occurs if user reviews errors and decides to fix an external error and wants to try the batch level	Manual	Migration Data Set Import - click Retry Objects		Migration Data Set Import - Retry Objects	44

Copyright © 2024, Oracle and/or its affiliates.

Step	Seq	Action	Manual / Batch	Portal - Action	Batch Control	Record Impacted - Resulting Status	Next Action Sequence
		Apply again at the object level.					
	42	**Apply Transactions**	Batch		F1-MGTAP - Migration Transaction Monitor - Apply	Migration Transaction - Applied or Error Applying Migration Object - Applied or Error Applying	43 Appropriate next action is based on error review, if applicable.
	43a	**Migration Data Set Import status update - auto transition to Completed** Applicable if all transactions are Applied	Batch		F1-MGDIM - Migration Data Set Import Monitor	Migration Data Set Import - Completed.	N/A
	43b	Migration Data Set Import status update - manual transition to Retry Objects. Occurs if user reviews errors and decides to fix an external error and wants to try the batch level Apply again at the Object level.	Manual	Migration Data Set Import - click Retry Objects		Migration Data Set Import - Retry Objects	44
	43c	Migration Data Set Import status update - manual transition to Retry Transactions. Occurs if user reviews errors and decides to fix an external error and wants to try the batch	Manual	Migration Data Set Import - click Retry Transactions		Migration Data Set Import - Retry Transactions	45

Copyright © 2024, Oracle and/or its affiliates.

Step	Seq	Action	Manual / Batch	Portal - Action	Batch Control	Record Impacted - Resulting Status	Next Action Sequence
		level Apply again at the Transaction level.					
	44	Migration Objects status update from Error Applying back to Approved. Occurs if user chose to Retry Objects.	Batch		F1-MGOPR - Migration Object Monitor [When using separate business data processes submit also F1-MGOPB (Migration Object Monitor (Business))]	Migration Object - Approved	
		Migration Data Set Import status update from Retry Objects to Apply Objects	Batch		F1-MGDIM - Migration Data Set Import Monitor	Migration Data Set Import - Apply Objects	40
	45	Migration Transactions status update from Error Applying back to Ready to Apply. Occurs if user chose to Retry Transactions or if the user transitions to Apply Transactions and the system detects that there are Transactions in the Error Applying state.	Batch		F1-MGTPR - Migration Transaction Monitor	Migration Transaction - Ready to Apply	42

Copyright © 2024, Oracle and/or its affiliates.

Step	Seq	Action	Manual / Batch	Portal - Action	Batch Control	Record Impacted - Resulting Status	Next Action Sequence
		Migration Data Set Import status update from Retry Objects to Apply Objects	Batch		F1-MGDIM - Migration Data Set Import Monitor	Migration Data Set Import - Apply Transactions	42

The following table summarizes the batch monitor jobs that are used in the import process. You can see that there are special monitor processes for the Apply step for both the Object and Transaction. However, for all other states that have monitor logic, the standard monitor process for that MO is used.

Batch Control	Description	Comments
F1-MGDIM	Migration Data Set Import Monitor	Processes data set records in the following states: • Pending • Ready to Compare • Apply Objects • Retry Objects • Apply Transactions • Retry Transactions
F1-MGTPR	Migration Transaction Monitor (Deferred)	Processes transaction records in the following states: • Pending • Error Applying
F1-MGTAP	Migration Transaction Monitor - Apply	Processes transaction records in the Ready to Apply state where the data set is in the Apply Transactions or Canceled state. **Note:** Be sure to set the number of threads to a number that does not exceed the number of threads supported by the thread pool worker.
F1-MGOPR	Migration Object Monitor	Processes object records in the following states: • Pending • Needs Review (check for Data Set cancellation) • Error Applying
F1-MGOPB	Migration Object Monitor (Business) Applicable only when separate business data processing is enabled.	Similar to F1-MGOPR (Migration Object Monitor) but restricted to business data migration objects.

Copyright © 2024, Oracle and/or its affiliates.

Batch Control	Description	Comments
F1-MGOAP	Migration Object Monitor - Apply	Processes object records in the Approved state where the data set is in the Apply Objects or Canceled state. **Note:** Be sure to set the number of threads to a number that does not exceed the number of threads supported by the thread pool worker.
F1-MGOAB	Migration Object Monitor (Business) - Apply Applicable only when separate business data processing is enabled.	Similar to F1-MGOAP (Migration Object Monitor - Apply) but restricted to business data migration objects.

Refer to Running Batch Jobs for more information about managing the batch jobs, including ways to automate the above steps.

Cancelling a Data Set

A user may choose to **Cancel** a data set to prevent it from being processed at any point during the process.

If related migration transactions or migration objects have already been created, they will not be canceled as part of the data set getting canceled (due to possible high volumes of related records). They will be canceled the next time an appropriate monitor batch process runs. The child records checks to see if the data set has been canceled prior to any state transition.

Additional Note Regarding Imports

The following points describe miscellaneous comments related to Migration Import.

- CMA relies on the fact that database referential integrity constraints are not in place, and that the SQL statements can be run in any order within the transaction. Any archiving solution that requires referential integrity constraints (such as Information Lifecycle Management) would not be possible on this data. Given that CMA migrations comprise administrative data and not transactional data, this should be a reasonable exception.
- The validation that is performed is only via the **Page Validate** service. BO validation algorithms are not executed. Page validation does not include validation of the business object against the schema (for example, for required fields, field sizes, etc.).
- If multiple migration requests are exported at the same time, on the import side, you should consider importing, reviewing, and applying an entire file/data set before moving on to the next one. The reason is that if objects are included in more than one file, two sets of "inserts" will be generated, but only the first will succeed. The second will cause the object to transition to "Cannot Apply". If instead you wait until the first file is completed before importing the second, the second data set will not generate any SQL for the object, since it has already been inserted. It's a matter of efficiency: If you first import all files and then try to apply all, you'll have to identify the duplicated object as an error and then mark the object as rejected before applying the transaction. This may also be avoided by using a Group migration request to include all objects in one file rather than multiple files.
- The system provides an algorithm to purge "unchanged" migration objects for a given migration data set. This may be plugged in as a BO exit algorithm on the **Ready to Compare** state for the Migration Data Set Import business object (**F1-MigrDataSetImport**).

The following points describe miscellaneous comments related to importing a business data set in bulk mode.

- The number of entities included in each migration object is captured on the migration object record.
- The comparison step only generates SQLs for new or changed entities. SQLs are not generated for unchanged entities.

Copyright © 2024, Oracle and/or its affiliates.

- Searching for migration objects based on their included entities is only supported for entities that were determined as new or changed. In other words, this type of search by included entities is only possible after the comparison step is performed.

Caching Considerations

There are two important caching considerations when using CMA to migrate administrative data.

- The "apply" batch processes update data. As described in Batch Cache, if any batch job is updating data marked as "cached for batch" it must be run in a special thread pool. Contact your system administrator and ask for the name of a thread pool that has turned off caching. This thread pool name must be entered in the batch controls used for CMA processing in the **Thread Pool Name** parameter.
- After a successful migration, the target region now has new administrative data which needs to be part of various caches. It is recommended to flush the server cache (which will trigger a 'global' flush of the cache). If the thread pool workers in the target region are configured to refresh their caches when a global flush is requested, then this is the only step required. If not, then the **F1-FLUSH** batch job should also be submitted to refresh the caches used in batch processing.

Fastpath: Refer to Caching Overview for more information.

Fastpath: Refer to Running Batch Jobs for more information about running CMA batch jobs.

Maintaining Import Data

This section describes the portals provided to add, view and maintain migration import data.

Migration Data Set Import

Use the Migration Data Set Import portal to view and maintain migration data set import records. Refer to Importing and Applying a Migration for an overview of the import process.

Navigate using **Admin** > **Implementation Tools** > **Migration Data Set Import.** You are brought to a query portal with options for searching for import data sets. In addition, the query provides an option to specifically search for data sets that have either objects in error or transactions in error. Once a data set has been selected, you are brought to the maintenance portal to view and maintain the selected record.

The following zones are visible on the **Main** tab:

- **Migration Data Set Import.** This zone contains display-only information about the selected record. Please see the zone's help text for information about this zone's fields.
- **Entity Summary.** This zone provides a summary count of all migration objects included in the current data set across the various statuses.

The following zones are visible on the **Transactions** tab:

- **Migration Data Set Transactions in Error.** This zone is only visible if there are transactions for this data set in a non-final state that have errors. It indicates the error for each transaction. A user may use this zone to review errors after the monitor batch job to apply transactions completes. The errors received when attempting to apply objects at the transaction level may differ from those received when attempting to apply objects at the object level. A transaction log is created for each object error received and these exceptions are shown in this zone.

  Note: Refer to Apply Step for more information about resolving errors.
- **Migration Data Set Transactions.** This zone is visible once the Import Step has occurred and lists all the transactions that are related to the data set. To see more information about a specific migration transaction, click the hypertext for its ID. This brings you to the Migration Transaction portal.

The following zones are visible on the **Objects** tab:

- **Migration Data Set Objects in Error.** This zone is only visible if there are objects for this data set in a non-final status that have errors. It indicates the error for each object. A user may use this zone to review errors after the

Copyright © 2024, Oracle and/or its affiliates.

monitor batch job to apply objects completes. Using the error information shown, the user can choose to drill into the record to transition it to **Error Applying** or choose to manually fix the cause of the errors and click **Retry Objects**. The user may also choose to select one or more records to **Reject**.

> **Note:** Refer to Apply Step for more information about resolving errors.

- **Migration Data Set Impacted Object Summary.** This zone is visible once the Import Step has occurred and lists the objects that are related to the data set. To see more information about a specific migration object, click the hypertext for its ID. This brings you to the Migration Object portal. A user may choose to update the status of one or more records by checking the records and clicking **Approve**, **Reject** or **Needs Review** accordingly.

Migration Transaction Portal

This page appears after drilling into a specific migration transaction from the migration data set portal or from the migration object portal.

Refer to Importing and Applying a Migration for an overview of the import process.

The following zones are visible on the main tab:

- **Migration Transaction.** This zone contains display-only information about the selected record. Please see the zone's help text for information about this zone's fields.
- **Migration Transaction Objects.** This zone lists the objects that are related to the data set. To see more information about a specific migration object, click the hypertext for its ID. This brings you to the Migration Object portal. A user may choose to update the status of one or more records by checking the records and clicking **Approve**, **Reject** or **Needs Review** accordingly.
- **Entity Summary.** This zone provides a summary count of all migration objects included in the current transaction across the various statuses.

Migration Object Portal

This page appears after drilling into a specific migration object from the migration data set portal or from the migration transaction portal.

Refer to Importing and Applying a Migration for an overview of the import process.

The **Migration Object** zone contains display-only information about the selected record. Please see the zone's help text for information about this zone's fields.

The **Migration Object - List of SQL Statements** zone lists the SQL statements that have been generated as a result of the comparison step. A check mark in the Suppressed checkbox indicates that the SQL for that row will not be applied during the apply step.

The **Migration Object - Algorithms** zone displays any pre-compare algorithms that are associated with the migration plan for this migration object.

Running Batch Jobs

There are several batch jobs that are part of the CMA process, especially the import step (which are highlighted in Import Process Summary). And in some cases, a single batch jobs may process multiple states in the same business object lifecycle. Implementations must decide the best way to manage the batch job submission depending on how they plan to work.

- **Batch scheduler**. If an implementation wishes to put these batch jobs in the batch scheduler, a given job may need to be included several times to manage progressing the records to completion.
- **Timed Batches**. The batch controls can be configured as timed batches so that they run every N minutes based on the setting. This allows for the batch jobs to run periodically and process whatever is ready. A user doesn't have to manually submit a batch request. Navigate to the Batch Control page and select the appropriate batch controls. For each one, change the Batch Control Type to **Timed**. Fill in the additional information that appears for timed batches.

Copyright © 2024, Oracle and/or its affiliates.

- **Event Driven**. The system provides BO enter plug-in algorithms and batch control post processing plug-in algorithms that automatically submit the appropriate next batch job for that step in the process. This allows for as much automation as possible for the steps that don't require user input. Note that configuration is required because the BOs / batch controls are not configured for this scenario by default. The following table highlights the BO and status where an algorithm may be plugged in and the name of the algorithm to use.

Business Object	Status	Algorithm
Migration Data Set Export	**Pending**	**F1-MGDPR-SJ** (Submit Migration Data Set Export Monitor).
Migration Data Set Import	**Pending**	**F1-MGDIM-SJ** (Submit Migration Data Set Import Monitor).
	Ready To Compare, **Retry Objects**	**F1-MGOPR-SJ** (Submit Migration Object Monitor). When using separate batch processes for business data add also this algorithm: **F1-MGOPRB-SJ** (Submit Migration Object Monitor (Business)).
	Apply Objects	**F1-MGOAP-SJ** (Submit Migration Object Apply Monitor). When using separate batch processes for business data add also this algorithm: **F1-MGOAPB-SJ** (Submit Migration Object Monitor (Business)).
	Apply Transactions	**F1-MGTAP-SJ** (Submit Migration Transaction Apply Monitor).
	Retry Transactions	**F1-MGTPR-SJ** (Submit Migration Transaction Monitor).

The following table highlights the batch controls where an algorithm may be plugged in and the name of the algorithm to use.

Batch Control	Algorithm
F1-MGTPR (Migration Transaction Monitor)	**F1-MGDIM-NJ** (Submit Migration Data Set Import Monitor).
F1-MGTAP (Migration Transaction Monitor - Apply)	**F1-MGDIM-NJ** (Submit Migration Data Set Import Monitor).
F1-MGOAP (Migration Object Monitor - Apply) When using separate batch processes for business data update also this batch control: F1-MGOAB (Migration Object Monitor (Business) - Apply)	**F1-MGDIM-NJ** (Submit Migration Data Set Import Monitor).
F1-MGOPR (Migration Object Monitor) When using separate batch processes for business data update also this batch control: F1-MGOPB (Migration Object Monitor (Business))	**F1-MGTPR-NJ** (Submit Migration Transaction Monitor).

Copyright © 2024, Oracle and/or its affiliates.

- **Manual submission**. The user managing the CMA import process submits the appropriate batch jobs on demand when a particular step is ready. Navigate to Batch Job Submission, select the appropriate batch control and fill in the parameters as needed.

Note that for the CMA batch processes used to migrate administrative data, there are multiple important caching considerations. Refer to Caching Considerations for more information.

CAUTION: Be sure that the Thread Count set when submitting the batch job does not exceed the number supported by the thread pool. Otherwise the extra threads will wait until the supported number of threads are finished, possibly resulting in a large number of errors in the Apply steps.

Refer to the parameter descriptions in the batch control metadata for more information about filling in the parameters.

For additional details on submission controls, refer to the topic Batch Job Submission - Main in the Batch Jobs section.

CMA Reference

This section provides additional reference information.

Framework-Provided Migration Configuration

This topic describes special information relating to migration objects provided for use by CMA in the product. Additional objects may be provided by your specific product. Any special information for objects is provided separately in each product's documentation.

The following points highlight some information about Framework-provided migration requests. Navigate to the migration request page in the application to view the details of all provided objects.

- Several base migration requests are supplied to logically group system and administrative tables. For example, there is a migration request for Framework System Configuration **F1-SystemConfig** where most system configuration objects are included. There is another one provided for CMA related configuration objects.
- The system supplies a group migration request **F1–FrameworkConfig** (Framework Configuration), which includes several other migration requests. The expectation is that this migration request includes all the typical objects that are included in a wholesale migration. Your specific product may include this migration request into its own group migration request to support a wholesale migration of all the framework and product administrative tables. An implementation may choose to build a custom group migration request. In this case, review the various migration requests provided by base to see if any may be included as components for the custom migration request. Then any new migration plans added to the base migration request in future releases are automatically included in future migrations.
- There are several different security related migration requests that include different combinations of migration plans to support multiple possible business requirements related to security migration. Note that the security migration request included in the above mentioned group migration request is the one that does not include users. If your implementation wants to copy users to a target environment, refer to Importing Data that References a User for some considerations.

Note: Refer to your specific product's CMA documentation for its recommendation on which migration requests to use for a full migration of framework and product administrative tables.

The following points highlight some information about the Framework-provided migration plans. Navigate to the migration plan page in the application to view the details of all provided objects.

- Fields and characteristic types are not migrated with an object (like a business object or a data area) unless specifically indicated.
- The **Application Service** used by an object is migrated only if it is CM-owned.

Copyright © 2024, Oracle and/or its affiliates.

- The **Batch Control** object optionally references a User (for 'timed' batches). Refer to Importing Data that References a User for some considerations about copying the user. Also note that when running a batch job, snapshot information is captured on the batch control. Updates like this increment the version number. If a batch control record is part of the migration and the comparison step has detected a change to the batch control, the Apply step will error out for this batch control if a batch job is submitted between the compare and apply step.

  Note: CMA batch controls that are part of the import step are executing and as such, the system does not include these records in a migration. If your implementation changes default parameters for any of the batch controls, the recommendation is to manually make those changes to the target region.

- The base migration plans for MO and BO include instructions to copy option types that use foreign key references to refer to other objects. Note that the data stored in the options are not validated, so defining these instructions is not required when doing wholesale migrations. However, including subordinate instructions for foreign key references is useful for targeted migrations to ensure that the related data is included in the migration. If you add additional MO or BO option types that use foreign keys and you want to support targeted migrations, you must create custom migration plans and requests for MO and BO, respectively to include these referenced objects in the migration plan. Note that you do not need to duplicate the instructions in the base migration plans. You may define the additional migration plans to only have the additional custom option types. When submitting a migration request for MO or BO you must include both the base migration plans and the custom migration plans in the request.
- For scripts, schema-based objects and zones, the migration plans provided by the product migrate, through constraints, some of the typical associated data with them. However, data specified through alternate formats (such as through **Edit Data** steps in scripts, referenced in schemas for schema-based objects, or data from mnemonics in zone parameters, etc.) are not identified and combined in the same transaction. The iterative processing functionality of the import step should resolve any timing issues that may result in validation errors for these types of objects.
- There are two migration plans for **Scripts**. The migration plan **F1-ScriptOnly** migrates just the script and its **Application Service** (provided the Application Service is CM-owned). The migration plan **F1-Script** includes most related objects, but does not migrate any objects referenced in the edit data area steps. It does not move the **Function** maintenance object. It may be included in any appropriate custom targeted migration request where scripts and related data should be migrated.
- If your implementation includes a **Feature Configuration** setting for the **F1_DBCONINFO** entry that will be included in a migration request, be sure that the import user on the target region has the appropriate security rights to this entry (**Administrator** access mode for the Feature Configuration application service (**CILTWSDP**).
- The common attachments in the Attachment maintenance object may be considered administrative data to include in a migration. Because this MO has a system generated key, as described in Administrative Data with System Generated Primary Keys, it uses a logical key of the file name and the creation date to determine if the record exists in the target environment. In addition, this MO contains administrative data (common attachments) and non-administrative data (owned attachments). To try to minimize the possibility of key "collision", new common attachments receive a generated key that includes a zero in the middle whereas owned attachments receive a generated key that does not have a zero in the middle.
- The Menu maintenance object has a user defined key, however, its menu lines and menu items have system generated keys. To avoid the possibility of overriding a menu line or menu item incorrectly, the menu MO will check the menu line's menu name in the source and target to be sure they match and will check the menu item's menu line in the source and target to be sure they match otherwise an error will be issued in the comparison step.
- For the system messages, the product provides three different migration plans.
  - Message Category and its Messages (F1-MessageCategory). This migration plan is included in the **F1-SystemConfig** migration request.
  - Message Category (F1-MessageCategoryOnly). This migration plan is provided to support a targeted migration where an implementation has created a custom message category and wants to move it but doesn't want to move all its messages.
  - Message (F1-Message). This migration plan is provided to support a targeted migration where only specific messages within a message category should be migrated.
- For lookup values, the product provides two different migration plans.
  - Lookup Field and its Values (F1-Lookup). This migration plan is included in the **F1-SystemConfig** migration request.

Copyright © 2024, Oracle and/or its affiliates.

  - Lookup Value (F1-LookupValue). This migration plan is provided to support a targeted migration where only specific lookup values within a lookup field should be migrated.
- There are some system data objects where no information in a base delivered record may be modified by an implementation. For these records, the base delivered migration requests include selection criteria to only select CM-owned records (because the base records will always exist in the target region assuming both regions have the same release). An example is Algorithm Type. The **F1-SystemConfig** migration request only includes CM-owned algorithm types. However, many system data objects support custom changes to one or more fields, for example the Zone object allows an implementation to override the zone text or certain parameters. Other system data objects support custom additions to a collection. For example, the Maintenance Object allows an implementation to add algorithms or options. For the migration plans related to these system data objects, all records are included in the base delivered migration requests to allow for any customized configuration to be migrated. It means that during the Import / Compare step many base delivered objects that are not customized will be marked **Unchanged**.
- Many of the integration related maintenance objects that include references to environment-specific data, such as Message Senders. This data should be migrated with extreme care. When appropriate, consider taking advantage of URI substitution. Refer to Referencing URIs for more information.

Copyright © 2024, Oracle and/or its affiliates.

Chapter 14

Facts

Topics:

- Fact Is A Generic Entity
- Fact's Business Object Controls Everything
- Fact Supports A Log

Fact is an optional configuration tool for simple workflow-type business messages or tasks. The base package does not provide a dedicated Fact user interface because fact is generic by design. Implementations configure their own user interface to visualize the desired custom business process. The topics in this section describe the generic Fact entity and how it can be customized.

Fact Is A Generic Entity

The Fact maintenance object is a generic entity that can be configured to represent custom entities and support automated workflows for a variety of applications. Each fact references a business object to describe the type of entity it is. A status column on the fact may be used to capture its current state in the processing lifecycle controlled by its business object.

The maintenance object also supports a standard characteristic collection as well as a CLOB element to capture additional information.

Fact's Business Object Controls Everything

A fact's business object controls its contents, lifecycle and various other business rules:

- Its schema defines where each piece of information resides on the physical Fact maintenance object.
- If may define a lifecycle for all fact instances of this type to follow. Each fact must exist in a valid state as per its business object's lifecycle definition.
- It may define validation and other business rules to control the behavior of facts of this type.

Fastpath: For more information about business objects, refer to The Big Picture of Business Objects.

Fact Supports A Log

The Fact maintenance object supports a log. Any significant event related to a Fact may be recorded on its log. The system automatically records a log record when the fact is created and when it transitions into a new state. In addition, any custom process or manual user activity can add log entries.

Fastpath:

Refer to State Transitions Are Audited for more information on logging.

Chapter 15

Conversion

Topics:

- Understanding The Conversion Process
- Conversion Entity Dashboard

This section describes the overall conversion process and the tools provided to support it.

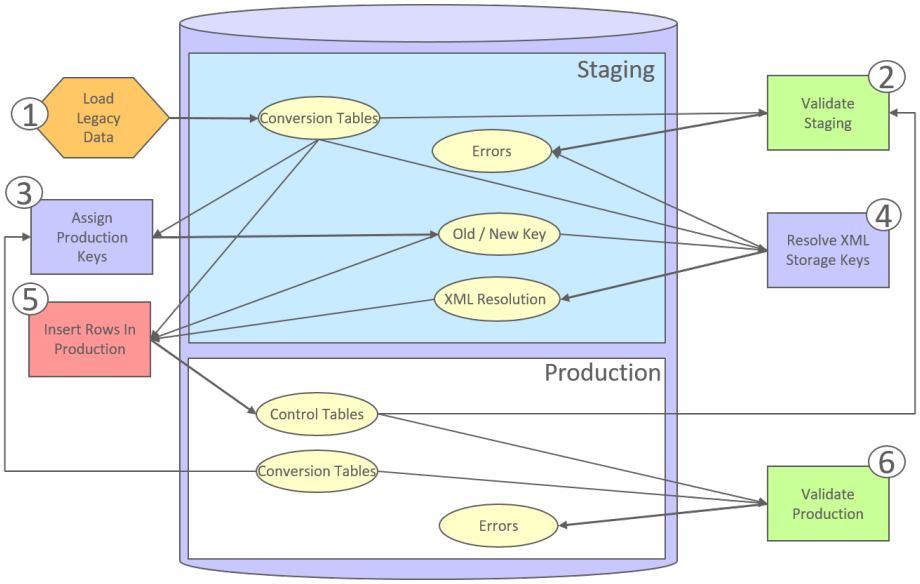
Understanding The Conversion Process

When you're ready to convert data from your legacy system, you will have analyzed your processing requirements according to your business and organizational needs and set up the control tables accordingly. After the control tables are set up, you are ready to load data into the system from your legacy system.

There are two table owners in the system database. We refer to the first owner as **staging** and the second owner as **production**. The staging owner is linked to the tables into which you insert your pre-validated data. The production owner is linked to the tables used by your production system.

Note: It's important to notice that **control tables exist only in production**. In staging, they exist as views to the production schema. Refer to Multiple Owners In A Single Database for more information about table ownership.

The conversion effort involves several steps as illustrated in the following diagram:



The following points briefly outline each of the above tasks:

- **Load Legacy Data.** During this step, your legacy data is loaded into the system. Notice that you are not migrating this data directly into production. Rather, your rows are loaded into tables that are identical to the production tables; they just have a different owner. Mapping legacy data into the system is probably the most challenging part of the conversion process because the system is a normalized database (and most legacy applications are not).
- **Validate Staging.** During this step, the system validates the data you loaded into the staging tables. Note that validation rules validate your staging data using the control tables that have been set up in production.
- **Assign Production Keys.** During this step, the system allocates random, clustered keys to the rows in the staging database.
- **XML Resolution.** During this step, the system resolves legacy keys that may be mapped to an XML storage field with new values that were assigned in the key assignment step.
- **Insert Rows Into Production.** During this step, the system populates your production tables with rows from the staging. When the rows are inserted, their prime keys and foreign keys are reassigned using the data populated in the key assignment and XML resolution steps.
- **Validate Production.** During this step, you rerun the object validation processes, but this time against production. We recommend performing this step to confirm that the insertion programs have executed successfully.

Note: Additional steps may be needed to support specific data conversion requirements. Refer to your product's documentation for additional conversion related information.

Copyright © 2024, Oracle and/or its affiliates.

This section further discusses concepts and guidelines related to the conversion process.

Conversion Entities

Conversion is enabled for selected master and transaction maintenance objects. In the same way, only a subset of a maintenance object's tables are enabled for conversion.

The following is provided for each maintenance object that supports conversion:

- An entity validation batch control. This batch process validates data in staging and can also be run to validate data in production.
- A key assignment batch control for each **table** that is eligible for conversion and has a system-generated key.
- An XML resolution batch control if the maintenance object has at least one table that includes an XML storage field.
- An insertion to production batch control for every **table** of the maintenance object that is eligible for conversion.

You may review the conversion configuration provided for an entity and more using the Conversion Entity Dashboard portal.

Conversion Steps

The following sections provide more details about the steps in the conversion process.

Note: Additional steps may be needed to support specific data conversion requirements. Refer to your product's documentation for additional conversion related information.

Load Legacy Data Into Staging Tables

This section provides some high level discussion about mapping and loading legacy data to the system's staging tables.

Mass Load Utility

Unless in a Cloud installation, you can use any method you prefer to load data from your legacy application. However, we recommend that you investigate your database's mass load utility (as opposed to using insert statements) as the mechanism to load the staging tables. In addition, we strongly recommend that you disable the indexes on these tables before populating these tables and then enable the indexes after populating these tables.

Note: In a Cloud installation, refer to the "Data Conversion Support for Cloud Implementations" for more information about the various tools provided to support database related conversion tasks.

Populating Keys

Some tables in the staging database have keys that are system-assigned random numbers. Special consideration should be given when populating these keys. Refer to A Note About Keys for more information on how to properly populate these tables. Those tables that don't have system-assigned random numbers have keys that are a concatenation of the parent's prime-key plus one or more additional fields.

Populating Characteristic Tables

There are many maintenance objects that include a characteristic table used to capture miscellaneous information about the object. Most of these tables include an indexed column used when searching by characteristic value called Search Characteristic Value. During conversion and depending on the type of characteristic, this column must be populated as follows:

- **Predefined.** Populate search characteristic value with the contents of the characteristic value column converted to upper case.

Copyright © 2024, Oracle and/or its affiliates.

- **Ad hoc.** Populate search characteristic value with the first 50 bytes of the ad hoc characteristic value column converted to upper case.
- **Foreign key.** Populate search characteristic value by concatenating the values of each foreign key characteristic value column to a maximum of 50 bytes.

Validate Information In The Staging Tables

During the first validation step, the system validates the data you loaded into the staging tables. This section provides some high level discussion about the validation process.

A Batch Process Per Maintenance Object

Each of the maintenance objects that are eligible for conversion must be validated using its respective entity validation conversion batch process. These are classic background processes that can also be run against production data.

These processes are multi-threaded and have no dependency on other processes.

Validation Mode

By default, each entity is fully validated for schema as well as entity validation rules, at the maintenance object and business object level. The batch process performs the same business rules that are used to validate data added by users in your production system. You may restrict the batch process to perform schema validation only or entity validation rules only using a batch parameter. This may be useful in identifying common data issues in preliminary runs.

Random Sample Mode

By default all records are selected for validation. The "Process Every Nth" batch parameter allows you execute this process in a random-sample mode to highlight pervasive errors. In this mode, you are actually telling the batch process to validate every N records.

We strongly recommend validating each entity in the following steps:

- Execute the validation batch process in random-sample mode.
- Review and correct the errors. Note, you can use the base package's transactions to correct an error if the error isn't so egregious that it prevents the object from being displayed on the browser.
- After all pervasive errors have been corrected; re-execute each object's validation batch process in all-instances mode to highlight elusive, one-off errors.

Reported Errors

Errors encountered during validation are logged onto the conversion Validation Error (CI_VAL_ERR) table. Note that at the start of this job, all rows in the conversion error table for the process maintenance object are deleted.

You can view errors highlighted by the validation process using the Conversion Entity Dashboard portal.

Recommendations To Speed Up Validation

The following points describe ways to accelerate the execution of the validation process:

- Ensure that statistics are recalculated after data has been inserted into the staging tables.
- Execute the process multi threaded.
- Execute shorter running validation processes (e.g., less records) first so that the error data can be analyzed while other processes are busy running.
- Remember that validation can be run in random-sample mode. We recommend running these batch processes using a large sample value for this parameter until the pervasive problems have been rectified.

Note: In a Cloud installation, refer to the “Data Conversion Support for Cloud Implementations” for more information about the various tools provided to support database related conversion tasks.

Copyright © 2024, Oracle and/or its affiliates.

Another use for these programs

In addition to validating your objects after conversion or an upgrade, the validation programs have another use. Say for example, you want to experiment with changing the validation of a business entity and you want to determine the impact of this new validation on your existing records. You could change the validation and then run that entity's validation batch process- it will produce errors for each record that fails the new validation.

Allocate Production Keys

During the key assignment step, the system assigns a new key for each legacy key for a table that its prime-key requires a system-generated random key. The conversion process allocates new prime keys to take advantage of the system's parallel processing and data-clustering techniques in the production system (these techniques are dependent on randomly assigned, clustered keys).

The topics in this section provides some high level discussion about the key assignment process and describe the background processes used to assign production keys to the staging data.

The Old Key / New Key Table

It's important to understand that the system does not overwrite the prime-keys on the rows in the staging database, as this is a very expensive IO transaction. Rather, a series of tables exist that hold each row's old key and the new key that will be assigned to it when the row is transferred into the production database. We refer to these tables as the "old key / new key" tables.

The convention "<1st letter of owner flag>K_<table_name>" is used to denote the old key / new key table name. For example, the old key / new key table for CI_ACCT is called CK_ACCT.

The insertion batch process that transfers the rows into the production database use the new key for the main record of the key along with any other record where this key is a foreign key. In the same way, the XML resolution process resolves conversion foreign keys residing in XML storage fields and replaces them with their corresponding new keys from these table.

A Batch Process Per Table

A key assignment batch process is provided for each table that has a system-generated key and belongs to a maintenance objects that is eligible for conversion. The batch process is responsible for populating the corresponding old key / new key table (i.e., you don't have to populate these tables). These processes are single threaded. Refer to a table's corresponding key assignment batch process for more information.

Key Assignment Dependencies

Most tables with system-generated keys do not inherit part of their key from another table's key. Their corresponding key assignment batch process have no dependencies and can therefore be executed in any order you please.

Some tables inherit part of their key from another table's key. Key assignment batch processes for such tables must be executed in key inheritance order. In other words, the key assignment process for a table should be run after the process that generates keys for the entity it depend on. The Conversion Entity Dashboard portal displays key inheritance dependency information if any exist for a conversion entity.

Note:

You may run multiple key assignment batch process in parallel as long as they are independent with respect to key inheritance.

Iterative Conversions

Rather than perform a "big bang" conversion (one where all records of an entity are populated at once), some implementations have the opportunity to go live on subsets of their entity base. If this describes your implementation, please be aware that the system takes into account the existing prime keys in the production database before it allocates

Copyright © 2024, Oracle and/or its affiliates.

a new key value. This means when you convert the next subset of customers, you can be assured of getting clean and unique keys.

Key assignment logic creates the initial values of keys by a manipulation of the sequential row number, starting from 1. After any conversion run, a subsequent conversion run will start with that row number again at 1, and the possibility of duplicate keys being assigned will be higher. The key assignment batch process allows you to specify a starting value for that row number. The purpose of this parameter is to increase the value of row number by the given value, and minimize the chance of duplicate key assignment. This parameter is only used if you are performing conversions where data already exists in the tables in the production database.

Run Type

Key assignment is performed in two steps:

- **Initial Key Generation.** The system allocates new keys to the rows in the staging tables (i.e., it populates the respective old key / new key table).
- **Duplicate Key Resolution.** The system reassigns keys that are duplicates compared to production.

By default, both steps are performed at the same run but you have the option to run them separately by indicating which step to run via a batch parameter. The proper use of this parameter will greatly speed up the key assignment step as described under **Recommendations To Speed Up Key Generation** section.

Recommendations To Speed Up Key Generation

The following points describe ways to accelerate the execution of the key generation programs.

- For a non-Cloud installation:
  - Make the size of your rollback segments large. The exact size is dependent on the number of rows involved in your conversion. Our research has shown that processing 7 million rows generates roughly 3GB of rollback information.
    - Setup the rollback segment(s) to about 10GB with auto extend to a maximum size of 20GB to determine the high water mark
    - A next extent value on the order of 100M should be used.
  - Make sure to turn off all small rollback segments (otherwise Oracle will use them rather than the large rollback segments described above).
  - After the key assignment programs execute, you can reclaim the space by:
    - Keep a low value for the "minimum extent" parameter for the rollback.
    - Shrink the rollback segments and the underlying data files at the end of the large batch jobs.
- Compute statistics on the old key / new key tables after every 50% increase in table size. Key generation is performed in tiers or steps because of the inheritance dependency between some tables and their keys. Although key generation for the inheritance dependency tier currently being processed is performed by means of set-based SQL, computation of statistics between tiers will allow the database to compute the optimum access path to the keys being inherited from the previous tier's generation run.
- Optimal use of the **Run Type** batch parameter.
  - Before any key assignments, alter both the "old key" F1_CX_ID index and the "new key" CI_ID index on the old key / new key tables to be unusable.
  - Run all key assignment batch processes in tiers, submitting each job to only perform the **Initial Key Generation** step.
  - Rebuild the indexes on the old key / new key tables. Rebuilding the indexes using both the PARALLEL and NOLOGGING parameters will speed the index creation process. Statistics should be computed for these indexes.
  - Run all key assignment batch processes in the current tier that were previously run in initial key generation mode, to perform the **Duplicate Key Resolution** step. This will reassign all duplicate keys.

Note: In a Cloud installation, refer to the “Data Conversion Support for Cloud Implementations” for more information about the various tools provided to support database related conversion tasks.

Copyright © 2024, Oracle and/or its affiliates.

XML Resolution

Most system-generated foreign keys are stored in physical fields and characteristic tables and as such their legacy value is replaced with its corresponding new key as part of the insertion to production process. Rarely, maintenance objects may store system-generated foreign keys in an XML storage field, i.e. a field defined with the data type of CLOB or XML. This step is only applicable to such maintenance objects.

During this step, the system resolves convertible system-generated foreign keys that may reside in XML storage fields you may have loaded into the staging tables. This section provides some high level discussion about the XML resolution process.

The XML Resolution Table

It's important to understand that the system does not overwrite the prime-keys on the XML storage fields in the staging database, as this is a very expensive IO transaction. Rather, a corresponding XML resolution table exists for each table that defines an XML resolution field to capture each row's resolved XML storage content, i.e. the content with all the old keys replaced with the new assigned keys.

The convention "<1st letter of owner flag>R_<table_name>" is used to denote the XML resolution table name.

The insertion batch process that transfers the rows into the production database replaces each XML storage field with its resolved value from the corresponding XML resolution table.

A Batch Process Per Maintenance Object

Each of the maintenance objects that are eligible for conversion and support XML storage fields is provided with an XML resolution batch process. These batch processes must be run to resolves foreign keys that may reside in these XML storage fields.

These processes are multi-threaded and must be executed after the key assignment step has completed and before inserting data to production.

XML Resolution Eligibility

Not all maintenance objects that support XML storage fields actually store convertible system-generated foreign keys in their XML storage field. If none of the business objects associated with the maintenance object involve mapping of such foreign keys to an XML storage field then XML resolution is not needed for any row in the maintenance object. The XML resolution batch process detects such situation and completes right away without storing any rows in any of the maintenance object's XML resolution tables.

Note: If the XML resolution batch process detects that the maintenance object is eligible, the maintenance object must reference a physical BO. If a product delivered business object supports system-generated foreign keys in an XML storage field, the product should be supplying a physical BO. If your implementation has system-generated foreign keys in an XML storage field in a custom business object for a maintenance object where the product has not supplied a physical BO, you must create a physical BO and link it to the maintenance object.

Only Resolved Values Are Captured

XML storage fields typically store large amounts of data. To avoid capturing the same XML content redundantly, the system only stores values in the resolved XML storage fields if the resolved value is different than the original value, i.e. at least one key was resolved.

If for a given record the resolved XML content is the same as the original content then the following rules apply:

- If this is the primary table of the maintenance object a record is inserted into its corresponding XML resolution table for that record with no value in the XML storage field.
- If this is a child table of the maintenance object then no record is inserted into the corresponding XML resolution for that record.

Copyright © 2024, Oracle and/or its affiliates.

Reported Errors

Errors encountered during XML resolution are logged onto the conversion Validation Error (CI_VAL_ERR) table. Note that at the start of this job, all rows in the conversion error table for the process maintenance object are deleted.

You can view errors highlighted by the XML resolution process using the "Validation Error Summary" page.

Insert Production Data

The topics in this section describe the background processes used to populate the production database with the information in the staging database.

A Batch Process Per Table

An insert to production batch process is provided for each table of a maintenance objects that is eligible for conversion.

The batch process is responsible for transferring to production all the rows from the staging table while replacing all references to old keys with their corresponding new keys as follows:

- Keys residing in prime keys, foreign key fields and characteristics are resolved using the corresponding old key / new key tables.
- Keys residing in generic foreign key reference tables where each row represents a reference to an entity's maintenance object and prime key values are also resolved using the corresponding old key / new key tables. Note that all entities referenced this way must either be convertible or view of production. A reference to an entity that is explicitly marked as not convertible is not supported as it cannot be resolved.
- Keys residing in XML storage fields are resolved using the corresponding XML resolution tables.

Note: Foreign keys residing in any other form of storage are not resolved. For example, any references stored in raw elements, log records, or any other field that is not used as an XML storage field.

All insertion batch processes are independent and may run concurrently. Also note, all insertion batch processes can be run in many parallel threads as described in the next section (in order to speed execution).

Recommendations To Speed Up Insertion

The following points describe ways to accelerate the execution of the insertion batch processes:

- Before running the first insertion batch process:
  - Rebuild the index on the prime key on the old key / new key table.
  - Re-analyze the statistics on the old key / new key table.
  - Alter all indexes on the production tables being inserted into to be unusable.
- After the insertion programs have populated production data, rebuild the indexes and compute statistics for these tables.

Note: In a Cloud installation, refer to the "Data Conversion Support for Cloud Implementations" for more information about the various tools provided to support database related conversion tasks.

Validate Production

During this step, you rerun the object validation batch processes, but this time in production. We recommend rerunning these batch processes to confirm that the insertion batch processes have executed successfully. We recommend running these batches in random sample mode (e.g., validate every 1000th object) rather than conducting a full validation in order to save time. However, if you have time, you should run these processes in full validation mode (to validate every object).

Copyright © 2024, Oracle and/or its affiliates.

A Note About Keys

The prime keys of the tables in the staging database are either system-assigned random numbers or they aren't. Those tables that don't have system-assigned random numbers have keys that are a concatenation of the parent's prime-key plus one or more additional fields. Every table whose prime key is a system-assigned random number has a related table that manages its keys; we refer to these secondary tables as "key tables".

The following points provide more information about the key tables:

- Key tables are used by programs that allocate new keys. For example, before a new key is allocated, the key assignment program checks the corresponding key table to see if it exists.
- Key tables only have two columns:
  - The key of the object.
  - An environment ID. The environment ID identifies the database in which the object resides.
- Key tables are named the same as their primary table with a suffix of "_K". For example: The key table for CI_ACCT is CI_ACCT_K.
- The key table for a table whose prime key is a system-assigned is defined on its table definition record.
- When you populate rows in tables with system-assigned keys, you must also populate a row in the related key table. For example, if you insert a row into CI_ACCT, you must also insert a row into CI_ACCT_K. The environment ID of these rows must be the same as the environment ID on this database's installation record.
- When you populate rows in tables that reference this record as a foreign key, you must use the appropriate key to ensure the proper data relationships. For example, if you insert a row in CI_SA for the above account, the ACCT_ID column must contain the temporary account key.
- When you insert rows into your staging database, the keys do not have to be random, system-assigned numbers. They just have to be unique. A later process, Allocate Production Keys, will allocate random, system-assigned keys prior to production being populated.

Multiple Owners In A Single Database

The conversion process relies on the following table owner configuration in the system database:

- The production owner is linked to the tables used by your production system. These tables have an owner ID of CISADM.
- The staging owner is linked to the tables into which you insert your pre-validated data. These tables have an owner ID of CISSTG.

The staging owner schema is almost identical to the production schema, with the following exceptions:

- Control tables are not actual tables in staging but rather views to the corresponding tables in production.
- Conversion specific tables designed solely to support the conversion process exist only in the staging schema. For example, tables that manage key assignment and XML resolution.

This section provides some high level concepts related to these table owners.

Validation Always Uses Production Control Data

When the validation batch processes run against your staging data, they validate the staging data against the production control tables (and insert errors into the staging error table). This means that the SQL statements that access / update entities needs to use the staging owner whereas the SQL statements that access control tables need to use the production owner. But notice that when these same validation batch processes run against production, the SQL statements will never access the staging owner. Rather, they all point at the production owner.

This is accomplished as follows:

- A separate application server must exist for each owner. Each application server points at a specific database user ID.

Copyright © 2024, Oracle and/or its affiliates.

> **Note:** In a Cloud installation, the application server may only point to a specific database user ID at any point in time. Refer to the "Data Conversion Support for Cloud Implementations" for more information about switching owners.

- The database user ID associated with the staging owner uses CISSTG as the owner for the master and transaction tables, but it uses CISADM as the owner of the production control tables.
- The database user ID associated with the production owner uses CISADM as the owner for all tables.

You may wonder why we went to this trouble. There are several reasons:

- We wanted to reuse the validation logic that exists in the programs that validate your production data. In order to achieve this, these programs must sometimes point at the staging owner, and other times they must point at the production owner (and this must be transparent to the programs otherwise two sets of SQL would be necessary).
- We wanted to let you use the application to look at and correct staging data. This can be accomplished by creating an application server that points at your staging database with the ownership characteristics described above.
- We wanted the validation programs to be able to validate your production data (in addition to your staging data). Why would you want to validate production data if only clean data can be added to production? Consider the following scenarios:
  - After an upgrade, you might want to validate your production data to ensure your pre-existing user-exit logic still works.
  - You may want to conduct an experiment of the ramifications of changing your validation logic. To do this, you could make a temporary change to user exit logic (in production) and then run the validation jobs on a random sample basis.
  - You forget to run a validation program before populating production and you want to see the damage. If you follow the instructions in this document, this should never happen. However, accidents happen. And if they do, at least there's a way to determine the ramifications.

Only Validation Can Work In Both Owners

While the redirection of owner ID's is a useful technique for the validation batch processes, it cannot be used by the key assignment and production insert batch processes? Why, because these processes have to access the same tables but with different owners at the same time. They also need to reference conversion specific tables that do not exist in production. For example, the batch process that inserts rows into a table in production must select rows from the staging version of that table, resolve keys from the conversion old key / new key mapping tables and insert the resolved records into the production version of that same table

This is accomplished as follows:

- In staging, a view to production exists for each eligible to conversion table. These views have hard-coded the database owner to point to production. For example, there is a view called CX_PER that points at person table in production.
- The key assignment and insertion programs use these views whenever then need to access production data.

Conversion Entity Dashboard

This portal provides an overview of a maintenance object's conversion configuration along with information related to its current status and progress within the overall conversion process.

Refer to Understanding Conversion for more information.

You can access the portal from the **Admin Menu** > **Implementation Tools** > **Conversion Entity Query.** You are brought to a query portal with options for searching for a specific maintenance object that is eligible for conversion. Once a maintenance object has been selected you are brought to the conversion entity dashboard portal to review the information for the selected record.

The following zones may appear as part of the portal's **Main** tab page.

- **Tables.** This zone provides information about tables that belong to the maintenance object and are marked as eligible for conversion.

Copyright © 2024, Oracle and/or its affiliates.

- **Batch Controls.** This zone lists the conversion batch controls defined for the maintenance object in the order they should be executed within the conversion process along with current execution details if any.
- **Error Summary.** This zone provides a summary of all conversion errors related to the maintenance object.
- **Duplicate Keys.** This zone lists duplicate legacy keys that need to be resolved by the key assignment conversion process.

The **Errors** tab page provides a query zone that allows you to review conversion errors related to the maintenance object by various criteria.

Copyright © 2024, Oracle and/or its affiliates.

Chapter 16

Data Export

Topics:

- Understanding Data Export
- Defining Data Export Control
- Generalized Export Dashboard

This section describes concepts and tools related to the various types of data export methods supported by the product.

Understanding Data Export

The application supports the following classes of export methods for a maintenance object:

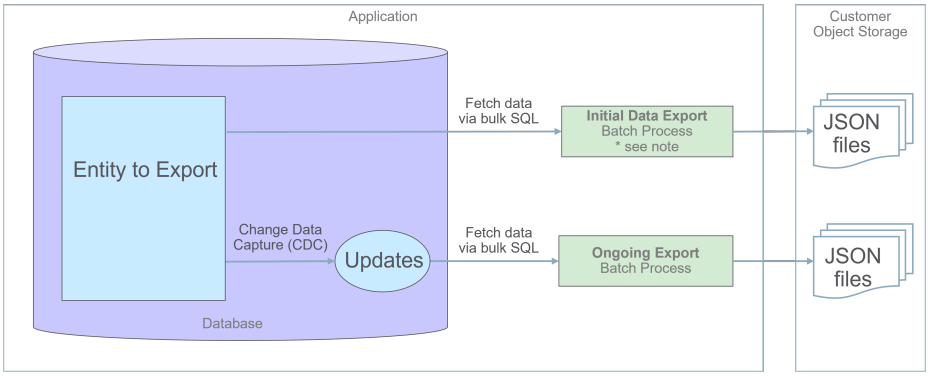
- **Generalized.** This method may be used for maintenance objects that are low to medium in volume and update activity. Most maintenance objects are eligible for this method.
- **Specialized.** This method should be used for maintenance objects that are high in volume or very frequently updated.

> **Note:** It is your organization's responsibility for designing the process by which these files are imported to one or more target environments.

This section discusses concepts and guidelines related to data export features.

Generalized Data Export

The generalized data export method provides a file-based export of data in JSON format. Exporting the data of a maintenance object using this method involves an initial export of the entity's entire data followed by an ongoing export of changes as they are made over time. These processes are illustrated in the following diagram:



The overall process involves the following steps:

- Enable entities for export in the Generalized Export Dashboard portal. You may enable many entities at once. This step also allows you to enable ongoing export of these entities.
- Submit the Generalized Initial Export Initiator (**F1-GEXPI**) batch process to export the current content of all enabled entities. This step submits a separate batch process for each entity you have enabled for export.

  > **Note:** A full database export can be used instead. The export file can be imported in the target environment to establish the initial baseline. Cloud customers may contact cloud operations to request such export for on-premise use. This schema export is limited for reporting purposes and is intended for initial extract only.

- Schedule the Generalized Ongoing Export (**F1-GEEXO**) batch process to run periodically.

The following sections further discusses concepts and guidelines related to the generalized data export method.

Eligibility

While most entities are eligible for generalized data export, not all of them are. Here are a few of the common reasons for that:

- The entity manages extremely high volume of data. Tracking changes to such entity would double the volume of managed data and impact overall performance.
- The entity is updated very frequently. Tracking these events may impact overall performance.

Copyright © 2024, Oracle and/or its affiliates.

- The entity is used by the product's infrastructure and operational processes. These entities are maintained differently and cannot be subject to the generalized data export methods.

By default a maintenance object is eligible for the generalized data export method unless explicitly marked otherwise by the **Data Export Class** option. The option may be used to mark an entity as either not allowed for any type of export or allowed for a specialized export only. The lack of such explicit option means the maintenance object is eligible for the generalized export.

CAUTION: If an entity is explicitly marked as not eligible for generalized data export there is a good reason for that and its configuration should not be changed.

One Method Fits All

The generalized data export method is designed to accommodate many eligible maintenance objects in a generic way that must also takes performance considerations into account. For this reason this method does not support entity specific filtering options nor custom rules of any kind.

The following rules apply to all entities that may be exported using this method:

- Entire data is exported.
  - All fields on all tables that belong to the maintenance object are exported except for "key" and "log" tables if any.
  - Maintenance object log tables typically do not contain useful business information and as such are omitted from export for performance and data volume reasons. However, there are entities for which log records provide analytical value. When applicable, use the **Export Log Tables** option to explicitly include log tables for a specific maintenance object.
- Same data structure and format is used for all entities. Refer to the **Record Format** section for more information.
- Data is exported to files only. Refer to the **Export To Files Only** section for more information.
- All changes are captured. It is not possible to exclude some changes from being tracked for ongoing export based on custom business rules.
- When data changes, a full snapshot of the data related to that entity is exported to avoid heavy data merging on the target side.
- When the entity is deleted, only the primary key of the entity is exported along with an indication that it is deleted.

Note: Export is not supported for entities that rely on the logical time zone functionality. This type of definition is not commonly used and includes maintenance objects that at least one of their fields is defined as stored in **Logical Standard Time**.

Export To Files Only

The immediate destination of both initial and ongoing export processes are files in a customer specified location. In a Cloud installation files are created in a customer owned Object Storage.

Based on your business requirements you may further consume these files by downstream applications such as data lakes etc.

Enabling Data Export

To enable data export for an eligible maintenance object a **Data Export Control** record needs to be created. The record keeps an indication of whether the initial export for the maintenance object has been completed or not and whether changes made to the entity should be tracked and exported on an ongoing basis. A designated portal allows you to enable the generalized data export for many maintenance objects at once as well as monitor their current export status at a glance. Refer to Generalized Export Dashboard portal for more information.

CAUTION: Enabling and disabling entities for export should only be made via the Generalized Export Dashboard or the Data Export Control portals to ensure the configuration is valid and complete.

Copyright © 2024, Oracle and/or its affiliates.

CAUTION: When applicable, ongoing data export should be enabled before submitting the initial export batch process for the entity. This ensures that changes made during the initial export are properly tracked.

Ongoing export options are cached in various online and batch caches. In order for these options to take effect and track entity changes made by any online or batch process, the various caches must be properly flushed when these options are changed. Using the designated portals to enable entities for export triggers a 'global' flush of the caches. If the batch thread pool workers are configured to refresh their caches when a global flush is requested, then this is the only step required. If not, then the **F1-FLUSH** batch job should also be submitted to refresh the caches used in batch processing. Refer to Caching Overview for more information.

Initial Data Export

Each maintenance object that is eligible for the generalized data export method is associated with an initial data export batch control that is referenced on the maintenance object using the **Export Batch Control** option. The same common batch program is used by all these batch controls to export the entire data of an entity to files. The process is multi-threaded and produces one file per thread by default. An optional batch parameter allows for multiple smaller files to be generated per thread by setting a limit on the number of records written to each file. Refer to the **File Size** section for more information.

Instead of manually submitting the generalized initial export batch process for each maintenance object you have enabled for export, submit the **F1-GEXPI** Generalized Initial Export Initiator batch process to automatically submit them all at once.

The initial export process for each maintenance object updates the corresponding data export control record for the maintenance object to indicate that the process has started and updates it again upon completion. This allows you to view the overall status of the initial export across all enabled entities on the Generalized Export Dashboard portal.

Note that the initiator batch process only submits batch processes for enabled maintenance objects that according to their data export control record their initial export has not started yet. If you need to export one or more entities again, reset the initial export status indication on their data export control records and submit the initiator batch process again.

The initial export file name is constructed as follows:

```
INIT_EXPORT[_file prefix(optional)]_[maintenance object]_[batch '
run]_[thread]_[thread count]_[timestamp][_file sequence].json.gz '
```

The "file sequence" part of the file name is used only when more than one file is created per thread. The sequence suffix is populated starting from the second file as follows _2.

Refer to any of the individual generalized initial export batch controls and the initiator batch control for more information.

Restricting Initial Export By Date

By default, the Initial Data Export batch process exports the entire data for an entity. In some situations, typically around high volume historical data, there may be a need to restrict the export to more recent data, for example the last few months of data.

The initial export batch process supports an optional batch parameter **Restrict By Date** that can be used to limit the scope of data to export. The parameter references a date field and a requested time frame in terms of number of days prior to business date. Refer to the **F1-GEIXP** Generalized Initial Export Template batch control for more information.

CAUTION: Excluding records from export should be carefully considered as this may result in data discrepancies when the excluded data is referenced on other exported entities.

Note: The option is only applicable to specific entities that have a qualifying reference date and the time frame for export may vary for each customer. Therefore, the new parameter is not added to any base product Initial Export batch control aside of the template batch control for documentation purposes. You may clone the base product batch control for a specific entity and add the parameter as per your business needs.

Copyright © 2024, Oracle and/or its affiliates.

Ongoing Data Export

Data changes are tracked by the Capture Change for Ongoing Export (**F1-MO-REGCHN**) Maintenance Object Audit algorithm. The algorithm records the prime key of a changed entity in a designated **Data Export Update** table that serves as a queue of changed entities to export. The queue is later on consumed by the ongoing data export batch process.

Note: The Maintenance Object Audit plug-in spot is capable of tracking changes made to the entity by any layer of the application. Making changes using direct SQL would bypass this mechanism and therefore should be avoided.

The audit rule is added to a maintenance object when the ongoing data export option on its corresponding data export control record is enabled. In the same way the audit rule is removed from the maintenance object when the ongoing export option is disabled.

Note: The ongoing export option has to be enabled on the data export control record for the entity before it is initially exported. This ensures changes made while the initial export process is in progress are not missed.

Unlike the initial export batch process that requires a batch control to be defined for each eligible maintenance object, a single batch process is used to export ongoing changes made to all entities.

The following describes the ongoing export batch process at a high level:

- All changes queued in the Data Export Update table are exported to a designated ongoing export file. Refer to the **Managing a Large Backlog** section for more information.
- A full snapshot of the data related to the changed entity is exported to avoid heavy data merging on the target side.
- Once exported, records are deleted from the Data Export Update table.

The process is multi-threaded and a separate file is produced for each thread by default. An optional batch parameter allows for multiple smaller files to be generated per thread by setting a limit on the number of records written to each file. Refer to the **File Size** section for more information.

Note: It is important to schedule the ongoing export batch process to run frequently enough to avoid accumulating a high volume of records in the Data Export Update table. The smaller the table is kept the faster it is for the change data capture mechanism to add records to it.

The ongoing export file name is constructed as follows:

```
INC_EXPORT_[batch run]_[thread]_[thread count]_[timestamp][_file sequence].json.gz
```

The "file sequence" part of the file name is used only when more than one file is created per thread. The sequence suffix is populated starting from the second file as follows _2.

Refer to Generalized Ongoing Export (**F1-GEEXO**) batch control for more information.

File Size

The initial export batch process produces a single file per thread by default. For some high volume entities this may result in files that are too large to manage. To produce multiple and smaller files for a specific entity, limit the number of records written to each file by setting the corresponding parameter on the initial export batch control defined for the entity. The same parameter may be provided when submitting the Initial Export Initiator batch process, in which case the initiator value is used only when a corresponding value has not been defined on the entity's specific batch control record.

In the same way, the ongoing export batch process produces a single file per thread by default. To produce multiple and smaller files, limit the number of records written to each file by setting the corresponding parameter on the batch control.

To avoid the creation of too many files by a single batch run, the system sets a maximum limit of around 500 files per run. When the requested number of records per file is determined to cause the number of generated files to exceed the system limit, the actual value used is adjusted to meet the system limit.

Copyright © 2024, Oracle and/or its affiliates.

Managing a Large Backlog

When a large backlog of changes accumulates in the ongoing export queue for some reason, the next batch takes longer to export which delays the downstream import step. A better option is for the backlog to be cleared in smaller "chunks", allowing the downstream process to import them as they are made available.

You may limit the number of records the batch process exports using the **Thread Processing Limit** batch parameter. When specified, the number of records exported by each thread is capped by the specified limit, leaving the remaining records to be consumed by subsequent batch runs. Instead of waiting for the next scheduled batch run, a new batch process may be automatically submitted, if explicitly requested to do so using the **Automatic Backlog Clearance** batch parameter.

It is important to ensure that the backlog clears gradually and that the queue resumes its normal size in a timely manner. The processing limit must therefore be set with an appropriate value that would promote proper clearance of the queue. To do so, at least 20% of the queue, and no less than 100,000 records, must be consumed by each batch run across all threads. If the specified limit does not meet the minimum clearance requirement then the processing limit is adjusted accordingly.

Record Format

Data is fetched and exported directly from the database in bulk read operations for performance reasons. For that reason, the structure used does not reflect the logical data model of the entity but rather its physical list of tables. Data for a given instance is organized by tables followed by rows in each table and for each row its list of fields in JSON format.

The following describes the format and structure used:

```
{
"OBJ": -"<mo name>",
"TIMESTAMP": -"<export time in ISO format for example 2019-07-25T11:06:04.740615Z>",
"PK1": -"<mo pk1 value>",
"PK2": -"<mo pk2 value if any>",  ← PK2-5 should only be included when applicable
"PK3": -"<mo pk3 value if any>",
"PK4": -"<mo pk4 value if any>",
"PK5": -"<mo pk5 value if any>",
"DELETED": true,  ← should only be included when the entity is deleted
"DATA": '
{
"<MO table name 1>":
[
{<name value pairs of all fields in row 1 in that table>},
{<name value pairs of all fields in row 2 in that table>},...
{<name value pairs of all fields in row n in that table>},...
],
"<MO table name 2>":
[
{<name value pairs of all fields in row 1 in that table>},
{<name value pairs of all fields in row 2 in that table>},...
{<name value pairs of all fields in row n in that table>}
],...
 -"<MO table name n>":
[
 {<name value pairs of all fields in row 1 in that table>},
{<name value pairs of all fields in row 2 in that table>},...
{<name value pairs of all fields in row n in that table>}
]
}
```

Notes about field values and formats:

- All fields for a row are included even when the field is empty or null.
- All string values are trimmed of excess trailing spaces but if a field is empty it exported as an empty string "" and not null.
- An empty array is used for a table node when no records exists for the entity in the table.
- Date/time information is converted to UTC time zone and exported in ISO format for example 2019-07-25T11:06:04.740615Z.

Copyright © 2024, Oracle and/or its affiliates.

- Deletes have an explicit indicator in the header.

Order of Consuming Files

It is important to apply all the initial export files for a maintenance object before applying ongoing change files to ensure data consistency.

The following is a recommended procedure for consuming generalized data export files:

- Apply all initial data export files before ongoing export files.
- Apply initial files for a maintenance object in run number order.
- Apply ongoing export files in run number order.
- If a new initial export is available the file consumption procedure should be changed temporarily as follows:
  - Stop consumption of ongoing export files.
  - Apply the new initial export files.
  - Resume consumption of ongoing export files.

Recover from a Lost Ongoing Export File

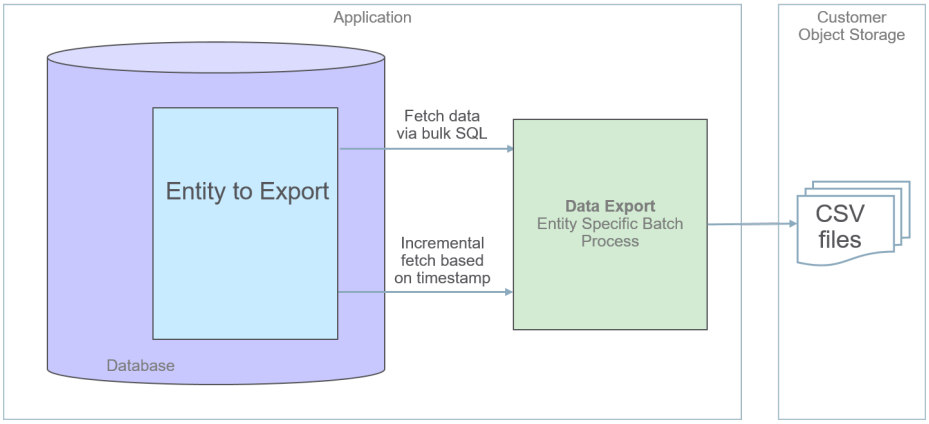
The ongoing export batch process keeps a backup of all entity keys, that were exported on a specific batch run, in a designated backup table. In rare situations where an ongoing export file is deleted or becomes corrupted before it gets processed, the backup table may be used to identify the missing changes and export them again.

Use the Restore Generalized Export Keys (**F1-GERST**) batch process to recover from this situation. The process identifies the entities that were exported during a specified batch run and adds them back to the ongoing changes queue from the backup table. The entities would then be exported again wen the next ongoing export runs. This method does not restore the exact content of the original file that was lost as it cannot be reproduced. Instead, the process ensures that the entities that were included in the lost file would be exported again.

Note: Backup records follow a weekly retention policy for performance reasons. They are meant to support near future situations of file recovery and are not an historical change data capture log.

Specialized Data Export

Several performance considerations are called for when exporting the data of a high volume entity or an entity that is updated in extremely high frequency. The generalized method does not fit such entities and a specific batch process is designed for each, aiming to achieve an optimal export method for the entity. The initial run of the process exports the entity's existing data while subsequent runs of the same process exports data incrementally, i.e. it exports records that were added or updated since the previous run based a timestamp field. The overall process is illustrated in the following diagram:



The following sections further discusses concepts and guidelines related to the specialized data export method.

Copyright © 2024, Oracle and/or its affiliates.

A Specific Process For Each Entity

A maintenance object that is designed to be exported using the specialized method is explicitly marked as such by the **Data Export Class** option and its specific batch process is defined using the **Export Batch Control** option.

All specialized batch processes are designed to export the data in a CSV format as it produces smaller files in size compared to more verbose formats. Refer to each batch process for more information about specific export considerations and parameters.

Enabling Data Export

To enable data export for a maintenance object a specialized type of Data Export Control record needs to be created. The record serves as an indication that such entity should be exported and keeps track of the date and time through which data has been exported so far, allowing subsequent export run to only export incremental changes.

Incremental Data Export

The specialized export method tracks changes incrementally from prior export run and does not use a change data capture mechanism. It therefore relies on a timestamp field on the primary table of the maintenance object to indicate when data was added or changed. Only entities that were added or changed since the previous export run are exported with each subsequent run. The date and time through which data has been exported is recorded for each maintenance object on its corresponding Data Export Control record.

Order of Consuming Files

Specialized data export files should be consumed in run number order to ensure a consistent and chronological uptake of data.

Defining Data Export Control

This portal is used to maintain Data Export Control records. These records capture your organization's export settings and current export state for a selected maintenance object. For a generalized export it keeps an indication as to whether the initial export for the maintenance object has been completed or not and whether changes made to the maintenance should be tracked and exported on an ongoing basis. For a specialized data export it keeps the date/time through which data was exported.

Refer to Understanding Data Export for more information.

You can access the portal from the**Admin Menu** > **Implementation Tools** > **Data Export Control.** You are brought to a query portal with options for searching for a specific record. Once a data export control record has been selected you are brought to the maintenance portal to view and maintain the selected record.

The **Data Export Control** zone provides information about the selected record.

Generalized Export Dashboard

The number of maintenance objects that are eligible for generalized export is large. This portal allows you to easily select and enable export for many eligible entities at once as well as review the current export state of those already enabled at a glance.

Refer to Generalized Data Export for more information about the generalized data export method.

You can access the portal from the **Admin Menu** > **Implementation Tools** > **Generalized Export Dashboard.**

The following zones may appear as part of the portal's **Main** tab page.

- **Export Entities.** This zone lists entities for which generalized data export control settings have already been set.

Copyright © 2024, Oracle and/or its affiliates.

- **Eligible Entities.** This zone lists entities that are eligible for the generalized data export method and for which such settings have not yet been set.

Copyright © 2024, Oracle and/or its affiliates.