



Telco e-Billing Manager (TBM) Developer's Guide

V4.5
Document ID: TBMG-08-4.5-01
Date Published: 9.29.04

© 1997–2004 edocs® Inc. All rights reserved.

edocs, Inc., One Apple Hill Dr., Natick, MA 01760

The information contained in this document is the confidential and proprietary information of edocs, Inc. and is subject to change without notice.

This material is protected by U.S. and international copyright laws. edocs is registered in the U.S. Patent and Trademark Office.

No part of this publication may be reproduced or transmitted in any form or by any means without the prior written permission of edocs, Inc.

edocs' Telco e-Billing Manager and edocs' Telco Application Suite, are trademarks of edocs, Inc. in the United States and other countries.

All other trademark, company, and product names used herein are trademarks of their respective companies.

Printed in the USA.

Table of Contents

Preface	9
1 TBM Functionality Overview	15
1.1 Telco e-Billing Manager Features	15
1.2 Telco e-Billing Manager Components	17
1.3 TBM Install Directories	17
1.4 TBM\Config Directory	19
1.5 About TBM Applications	19
1.6 Application Directories	20
1.7 Application Profiles	21
1.8 Application Packaging	21
2 Presentation Framework	31
2.1 Introducing TBM Application Basics	31
2.1.1 TBM Application Fundamentals	31
2.1.2 Architecture & Structure	32
2.1.3 Standard Page Flows	33
2.2 Struts and Tiles and the edocs View Architecture	33
2.2.1 TBM Application Architecture	34
2.2.2 TBM Struts and Tiles	34
2.2.3 Mapping DDNs	36
2.2.4 Struts Extensions	36
2.2.5 TBM Tiles	36
2.2.6 View Tiles	39
2.2.7 Creating Custom Tabs	40
2.3 Development Utilities	44
2.3.1 TBM Tags	44
2.3.2 Working With Properties	46
2.3.3 Working With Logs	47
2.3.4 Developing Custom Actions	48
2.3.5 Customer Account Manager	48
2.4 Look and Feel	49
2.4.1 Message Rendering and Localization	50
2.4.2 Style Sheets	51
2.4.3 Images	51
3 Statementing and Content Access	53
3.1 View Architecture Basics	53
3.2 Downloadable View Support	54
3.3 Writing and Configuring Post Processors	55

3.3.1	How to Write a Post Processor	56
3.3.2	Pattern Processing	59
3.4	Live Data Extraction	60
3.4.1	About the LDE Manager	61
3.4.2	About LDE Data Streams	62
3.4.3	Developing Live Data Extractors	62
4	Security: Enrollment, Authentication, Authorization.....	69
4.1	About Role-Based Security.....	69
4.1.1	What is Security and Authorization Framework (SAF)?	70
4.1.2	Authenticate.....	70
4.1.3	Security Profile	71
4.1.4	User Profiles	71
4.1.5	Security Management.....	71
4.1.6	Authorization.....	71
4.2	Controlling Access to Resources	72
4.2.1	Using the SAF Sub-system	72
4.2.2	SAF Components	72
4.2.3	Policy Store	72
4.2.4	Rights Engine	73
4.2.5	Implied Policies.....	74
4.2.6	Permission Classes	74
4.2.7	Custom Impliers.....	74
4.2.8	JSP Permission - Example	74
4.2.9	Using programmatic security.....	75
4.3	Integrating Security Providers.....	77
4.3.1	BSL Security SPI's	77
4.3.2	BSL Security Concepts.....	78
4.3.3	Common Implementation Use Cases.....	78
4.3.4	Authentication	79
4.3.5	Security Manager Profiles	79
4.3.6	User Profiles	80
4.3.7	Configuring BSL Providers	80
5	Customer Service Representative (CSR) Access/Capabilities	83
5.1	Overview	83
5.2	CSR Access (Impersonate User).....	83
5.3	CSR Application	85
6	Payment Cartridges	87
7	Downloading Views and Converting Advanced Function Print Files (AFP) to PDF.....	89
7.1	Downloading CSV, XML, and PDF Views	89
7.2	AFP-PDF Conversion	104
8	BSL Hierarchy Connector	105
8.1	Overview	105
8.2	How to Create and Manage Hierarchies via XML	106
8.3	How to Synchronize Hierarchies with eStatement Indexer Data	112

8.4 Hierarchy APIs.....	115
8.4.1 Creating Hierarchies Programmatically.....	116
8.4.2 Adding Entities to a Hierarchy	116
8.4.3 Finding Hierarchies, Nodes, and Folders	117
8.4.4 Updating Nodes.....	118
8.4.5 Giving User Access to Nodes.....	118
8.5 How to Implement Hierarchy-Based Access Control (HBAC).....	118
8.6 How to Search for and Find Objects Within Hierarchies.....	119
9 Profile Management.....	121
9.1 Overview	121
9.2 How to Manage the User Profile	121
9.3 How to Manage the Security Profile.....	121
9.4 How to Manage the Company and Company User Profiles	122
10 Notifications	123
10.1 Notification Setup UI.....	123
10.2 Notification Setup Actions.....	125
10.3 Messenger Component	128
10.4 How to Write Custom Jobs for Email Notification	130
11 Address Books.....	135
11.1 Address Book Overview	135
11.2 Personal Address Books	135
11.3 Corporate Address Books.....	142
11.4 AddressBook Component.....	143
11.4.1 Displaying the Address Book	144
11.4.2 Showing Addresses.....	146
11.4.3 Saving and Deleting Addresses	146
11.4.4 Address Book View Processing	146
11.4.5 Address Book Post-Processing of TBM Views.....	146
12 Custom Jobs	149
12.1 About Jobs and the Shell Command Task	149
12.2 Defining a New Job Type.....	150
12.2.1 Create the Job Type Script.....	150
12.2.2 Configuring Your New Job Type	157
12.2.3 Another Example of Defining a New Job Type	158
13 Charting	161
13.1 Introduction and Components.....	161
13.1.1 Charting Use Case Overview Diagram	162
13.1.2 Components of Charting.....	163
13.2 Configuring Charting for Your Server.....	163
13.2.1 About Servers and Charting	163
13.2.2 About Fonts	163
13.2.3 Configuration Activity Diagram	164
13.2.4 Setting Display Devices and xvfb.....	164
13.2.5 Setting Display Permissions and xhost	165
13.2.6 Setting Display Awareness.....	166
13.2.7 Configuring a Headless Server for Charting.....	166

13.3 Composing Charts in Statements	167
13.3.1 About Charting in the Composer	167
13.3.2 Inserting a Chart Tag in the Composer	168
13.3.3 Naming Conventions for Charts	170
13.3.4 About Chart Tags in the ALF	170
13.3.5 About The Chart Properties File	171
13.3.6 About Simulating Charts	171
13.4 Customizing Chart Properties	172
13.4.1 About Chart Attributes in the ALF	172
13.4.2 Customizing a Chart in the ALF	173
13.4.3 Customizing the Chart Properties File	174
13.4.4 Chart Type	174
13.4.5 Pie	175
13.4.6 Bar	177
13.4.7 Line	177
13.4.8 Horizontal Bars (HorizBar)	178
13.4.9 Individually Colored Bars (IndBar)	178
13.4.10 Individually Colored Columns (IndColumn)	178
13.4.11 (LabelLine)	179
13.4.12 Polar Chart (Polar)	179
13.4.13 Regression (Regress)	179
13.4.14 Stacked Bars (StackBar)	180
13.4.15 Stack Column Chart (StackColumn)	180
13.4.16 Stick Chart (Stick)	180
13.4.17 StickBar Chart (StickBar)	181
13.4.18 General Properties	181
13.4.19 Background Properties	181
13.4.20 Plot Area Properties	182
13.4.21 Axis Properties	183
13.4.22 Legend Properties	186
13.4.23 Favorite Colors Properties	187
13.4.24 Default Chart Properties	187
13.4.25 Customizing Default Properties	190
13.4.26 Previewing Charts with com.edocs.app.chart.Simulator	191
13.5 Publishing Charts	192
13.5.1 Before Publishing Charts	192
13.5.2 Publishing a Chart View	193
13.5.3 Viewing Charts in Statements	194
13.6 Designing Custom Charts with the Charting Servlet	195
13.6.1 Customizing Charter.java	195
13.7 Troubleshooting Charts	198
13.7.1 Charting Checklist	198
13.7.2 Common Problems and Known Issues	198
13.8 Application Programming Interfaces (APIs) for Charting	200
13.8.1 Package com.edocs.app.chart Description	200

13.8.2 Class ChartClient.....	200
13.8.3 Class ChartData	202
13.8.4 Class Charter.....	203
13.8.5 Class Constants	204
13.8.6 Class PublisherCommon.....	204
13.8.7 Class PublisherWrapper.....	205
13.8.8 Class Simulator	205
13.8.9 Class Util	205
13.9 Default Properties and Attributes	206
13.9.1 ChartDefaults.properties.....	206
13.9.2 NW_LocSummary.ALF	216

About Telco e-Billing Manager (TBM)

The objective of the B2B TBM application is to provide online account management and customer self-service for telecommunications provider's business customers.

The main features of B2B are:

- **Administration** – Online management of company profiles and company users, hierarchy management, and managing business structures and corporate address books.
- **Statements** – Telco's customers can view summary and detailed statements on accounts, devices, and usage.
- **Analytics** – Standard account, device, and usage reporting functionality. Advanced reporting and report creation is available in a separate application (TAM).
- **Service Management** – A separate application (TSM) for service management and analysis is available for integration into TBM.
- **Payments** – Companies can set up payment accounts for recurring and one-time payments, which historical payment activity for a configurable period of time.
- **User Profile Management** – Users are able to manage personal profiles and address books, user names and passwords, and notifications.
- **CSR Management** – edocs' standalone application for CSR management. Includes searching accounts and invoking impersonate company user functionality.

About This Guide

The edocs Software Developers Kit allows developers to write custom code against edocs applications. This SDK guide is intended for edocs system integrator partners, senior developers with an edocs client company, and edocs Professional Services representatives.

The TBM SDK assumes you have an in-depth understanding of and practical experience with:

- TBM system architecture, installation, deployment, application design, and administration
- Java 2 Enterprise Edition (J2EE), Enterprise JavaBeans (EJBs), servlets, and JSPs

- Packaging and deploying J2EE applications for WebLogic or WebSphere
- Directory services including the Java Naming Directory Interface (JNDI) and the Lightweight Directory Access Protocol (LDAP)
- HTML and XML, web server administration, and web browsers

This guide also assumes you have:

- Read the TBM product documentation and are familiar with TBM functionality
- Read the javadoc that comes with the TBM SDK
- Successfully installed TBM in a J2EE development environment
- Knowledge of how to develop J2EE web applications using JSP, Struts, Tiles and XML

Related Documentation

This guide is part of the TBM documentation set. For more information about implementing your TBM application, see one of the following guides:

Print Document	Description
<i>TBM Installation Guides</i>	How to install Telco e-Billing Manager in a distributed environment.
<i>TBM Presentation Design Guide</i>	How to use Composer to define the rules for mapping data to templates for viewing statements.
<i>TBM Administration Guide</i>	How to set up and run a live TBM application in a J2EE environment.
<i>TBM Data Definition Guide</i>	How to use DefTool to define the rules for data extraction in a DDF file.
<i>TBM Presentation Design Guide</i>	How to design data presentment for a TBM application with the Composer.

The following online help is also available for the TBM Command Center and tools (DefTool and Composer).

Online	How to Access
DefTool and Composer Help	In DefTool or Composer, select Help>Help Topics .
Command Center Help	In the Command Center, click the Help button on the menu. You can also click the Help button on Command Center screens for context-sensitive help.

Obtaining edocs Software and Documentation

You can download edocs software and documentation directly from Customer Central at <https://support.edocs.com>. After you log in, click on the Downloads button on the left. When the next page appears, you will see a table displaying all of the available downloads. To search for specific items, select the Version and/or Category and click the Search Downloads button. If you download software, an email from edocs Technical Support will automatically be sent to you (the registered owner) with your license key information.

If you received an edocs product installation CD, load it on your system and navigate from its root directory to the folder where the software installer resides for your operating system. You can run the installer from that location, or you can copy it to your file system and run it from there. The product documentation included with your CD is in the Documentation folder located in the root directory. The license key information for the products on the CD is included with the package materials shipped with the CD.

If You Need Help

Technical Support is available to customers who have an active maintenance and support contract with edocs. Technical Support engineers can help you install, configure, and maintain your edocs application.

This guide contains general troubleshooting guidelines intended to empower you to resolve problems on your own. If you are still unable to identify and correct an issue, contact Technical Support for assistance.

Information to provide

Before contacting edocs Technical Support, try resolving the problem yourself using the information provided in this guide. If you cannot resolve the issue on your own, be sure to gather the following information and have it handy when you contact technical support. This will enable your edocs support engineer to more quickly assess your problem and get you back up and running more quickly.

Please be prepared to provide Technical Support the following information:

Contact information:

- Your name and role in your organization.
- Your company's name
- Your phone number and best times to call you
- Your e-mail address

Product and platform:

- In which edocs product did the problem occur?
- What version of the product do you have?
- What is your operating system version? RDBMS? Other platform information?

Specific details about your problem:

- Did your system crash or hang?
- What system activity was taking place when the problem occurred?
- Did the system generate a screen error message? If so, please send us that message. (Type the error text or press the Print Screen button and paste the screen into your email.)
- Did the system write information to a log? If so, please send us that file. For more information, see the *TBM Troubleshooting Guide*.
- How did the system respond to the error?
- What steps have you taken to attempt to resolve the problem?
- What other information would we need to have (supporting data files, steps we'd need to take) to replicate the problem or error?

Problem severity:

- Clearly communicate the impact of the case (Severity I, II, III, IV) as well as the Priority (Urgent, High, Medium, Low, No Rush).
- Specify whether the problem occurred in a production or test environment.

Contacting edocs Technical Support

You can contact Technical Support online, by email, or by telephone.

edocs provides global Technical Support services from the following Support Centers:

US Support Center

Natick, MA

Mon-Fri 8:30am – 8:00pm US EST

Telephone: 508-652-8400

Europe Support Center

London, United Kingdom

Mon-Fri 9:00am – 5:00 GMT

Telephone: +44 20 8956 2673

Asia Pac Rim Support Center

Melbourne, Australia

Mon-Fri 9:00am – 5:00pm AU

Telephone: +61 3 9909 7301

Customer Central<https://support.edocs.com>**Email Support**<mailto:support@edocs.com>

Escalation process

edocs managerial escalation ensures that critical problems are properly managed through resolution including aligning proper resources and providing notification and frequent status reports to the client.

edocs escalation process has two tiers:

1. **Technical Escalation** - edocs technical escalation chain ensures access to the right technical resources to determine the best course of action.
2. **Managerial Escalation** - All severity 1 cases are immediately brought to the attention of the Technical Support Manager, who can align the necessary resources for resolution. Our escalation process ensures that critical problems are properly managed to resolution, and that clients as well as edocs executive management receive notification and frequent status reports.

By separating their tasks, the technical resources remain 100% focused on resolving the problem while the Support Manager handles communication and status.

To escalate your case, ask the Technical Support Engineer to:

1. Raise the severity level classification
2. Put you in contact with the Technical Support Escalation Manager
3. Request that the Director of Technical Support arrange a conference call with the Vice President of Services
4. Contact VP of Services directly if you are still in need of more immediate assistance.



TBM Functionality Overview

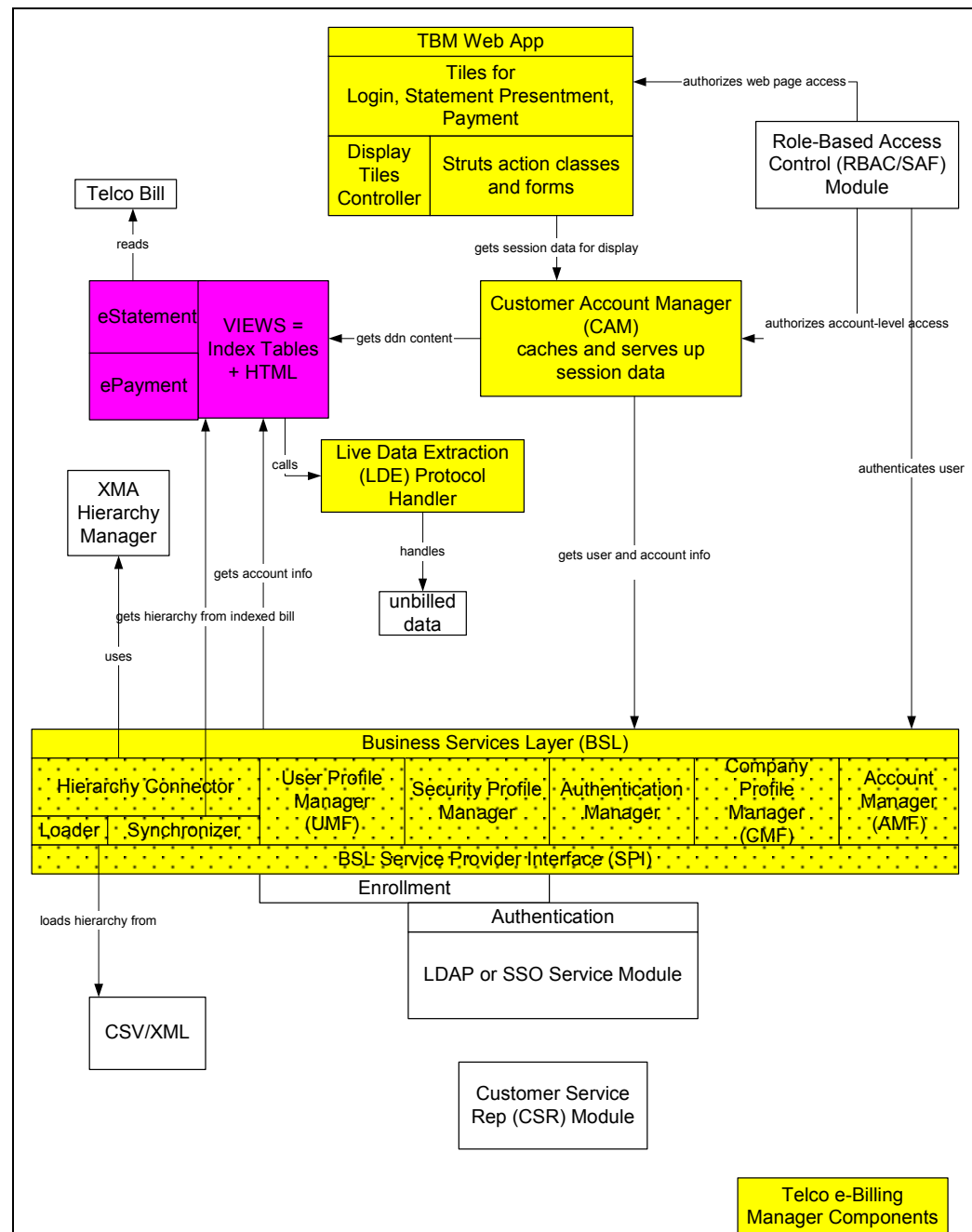
1.1 Telco e-Billing Manager Features

The following table provides a complete list of the use cases that specify the functionality of the TBM application:

Requirement Category	Description	Use Cases
Enrollment and Login	Enroll users and manage the login user session. The <i>Log In To Application</i> use cases is a pre-condition for all customer use cases	Login to Application Logout of Application Enroll Company Users Enroll As User Forgot Password
Administration	Manage the billing hierarchy.	Manage Company Profile Manage Billing Accounts (3 use cases) Manage Billing Account Users Manage Business Structures (4 use cases) Manage Business Structure Users Manage Business Structure Devices Edit Business Structure Devices or Accounts Manage Folder Nodes (4 use cases) Manage Company Users (2 use cases) Manage Corporate Address Book
Statement Presentment	Provide a place to perform functions related to presenting and accessing customer statements	View Business Dashboard View Account Summary View Device List View Device Summary View Record Details for Device View Unbilled Activity

Requirement Category	Description	Use Cases
		View Bill Messages Print Current View Print Invoice Download Account Data
Payment	Handle user payments	Make One-Time Payment Setup Recurring Payment Manage Payment Accounts View Payment Activity
Analytics	Summarize data in a variety of reports	View Account Level Report View Device Report
Profile Management	Manage user account and profile information Send e-mail notifications as appropriate to customer	Manage Personal Profile Manage Notifications Manage Personal Address Book (PAB) Manage PAB Entry Manage Password Manage Username
Customer Service	Customer Service Application that enables critical account management access and capabilities to customer service representatives.	Create Organization Create Organization Summary Create Administrator User Create Administrator Summary Edit Administrator Edit Administrator Details Summary Search Organization Search Organization Results View Accounts for Organization View Administrators for Organization Search for an Account Search Account Results View Last Statement for Account Add Billing Account Billing Account Added Search for Internal User Search Internal User Results Edit Internal Users Edit Internal Users Summary Add Internal Users Add Internal User Summary Edit Internal User Profile
Help		Telco-specific (FAQs, Contact, Terms, Conditions)

1.2 Telco e-Billing Manager Components



1.3 TBM Install Directories

Folder	Contents
Main TBM Folder	Root Directory (c : \edocs or /opt/edocs)

Common	SDK Support default files
Config	Configuration common all of TBM
DB	Schema files for Oracle, DB2 and MSSQL
Estatement	Statement specific support
J2EE apps	Command Center and TBM default applications
Jre	JRE Support
Lib	Common libraries
payment	Payment specific support
Release notes	Release notes
Uninstall	Uninstall Support
views	Prepackaged views

The J2EEApps directory contains:

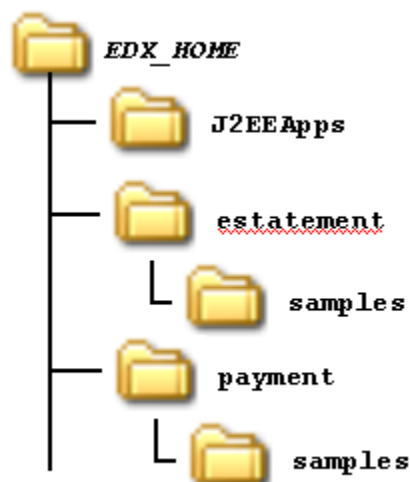
- eStatement - the edocs command center application
- tbm-b2b (Business Edition) - TBM b2b out of the box application
- tbm-csr (Business Edition) - TBM CSR Application
- tbm-b2c (Consumer Edition) -TBM b2c out of the box application

The estatement directory contains:

- Support, samples and documentation core to statement based applications

The payment directory contains:

- support, samples and documentation core to payment based applications



1.4 TBM\Config Directory

The TBM\Config directory contains:

- `app-config.properties` - Properties specific to all TBM applications
- `common-logging.properties` - Apache logging properties
- `hibernate.properties` - Hibernate properties for Oracle,DB2 and MS-SQL
- `hierarchy.cfg.xml` - Support for hierarchy
- `hierarchy.hibernate.properties` - Hibernate properties specific to hierarchy
- `log4j.xml` - Logging properties

1.5 About TBM Applications

TBM Applications are combination of:

- A set of artifacts created and managed by the edocs Command Center
- J2EE Applications
- A set of application DDN and other properties associated with an application (run time configuration via `app-config.properties`)

You create and name a TBM application in the Command Center to match the APP_DDNN entry in `app-config.properties`. This DDN becomes the parent or primary ddnn associated with the application.

By default TBM applications contain two DDNN entries:

- DDNN.Summary - which specifies the source of the summary data
- DDNN.TBM - which specifies where core TBM data will be obtained

While these two DDNNs are often the same, both referring to the actual DDNN TBM, they can use different data sources for content.

The "primary" application created with Command Center must match the APP_DDNN entry in `app-config.properties`:

```
FILE_PROP_VERSION=TIME_STAMP
```

```
APP_DDNN=TBM
```

```
# DDNN
```

```
DDNN.Summary = TBM
```

```
DDNN.TBM=TBM
```

```
DDNN.Unbilled=Unbilled
```

```
# Views
DDN_NAME_AccountList = Account_Summary

#LDE_MAX_CACHE_SIZE=52428800
. . .
```

1.6 Application Directories

Artifacts of the Command Center create application operation are:

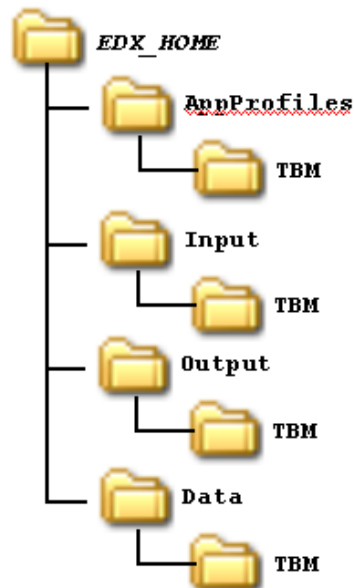
- Registration of the application
- Creation of AppProfile, input, data and output directories

When you use the Command Center to create TBM application, it creates a number of directory artifacts. These artifacts represent Input, Data and Output processing directories, each named based on the application. Indexer and other jobs use these directories to manage files.

As files are processed time stamps are prepended to the file name to allow for intelligent management of multiple instances of the same file name.

Indexer jobs:

- Process files found in Input directories
- Move results to Data directories

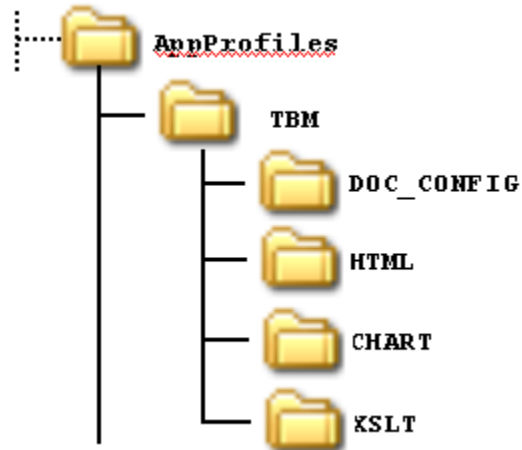


1.7 Application Profiles

Application profiles:

- Are created when you add a new application in Command Center
- Contain all the underlying artifacts of publishing
- Contain subdirectories by functional area (HTML, Indexer support etc)

Functional areas contain timestamped directories containing "version sets"



1.8 Application Packaging

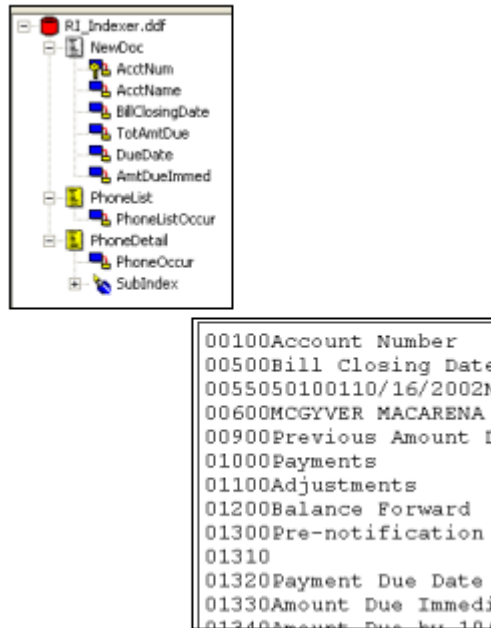
Telco e-Billing Manager applications:

- Are packaged as an Enterprise Application Archives (.EARS)
- Contain a set of supporting EJBs
- Contain one or more Web Applications (.WARs)
- Must have an associated a Data Definition Name (DDN) created and configured using the Command Center, including all views, application logic files, html templates and data definitions required for presentment

Data Definition Files:

- End with a .ddf extension
- Contain the rules for finding and extracting customer data with a "paper" bill
- Are created using representative business data and edocs tools

Specialized DDF files are used for a core indexer batch job run regularly to prepare data for presentation:



Views, application logic files, and HTML templates

Views:

- Are combinations of data definitions, HTML templates and data created with edocs tools
- Are used to present client data
- Can be formatted to generate HTML, XML, Charts or other formats such as CVS
- Are versioned

Application Logic Files:

- Contain the rules to present extracted data
- Define formatting, sorting, filtering, business logic and output formatting.
- May use HTML templates

A view is a set of design files what result in particular presentation of statement data. View files enable a user to dynamically display formatted statements, review notifications, emails or other account data.

A typical HTML based statement view includes a pair of DDF and ALF files as well as one or more template HTML files.

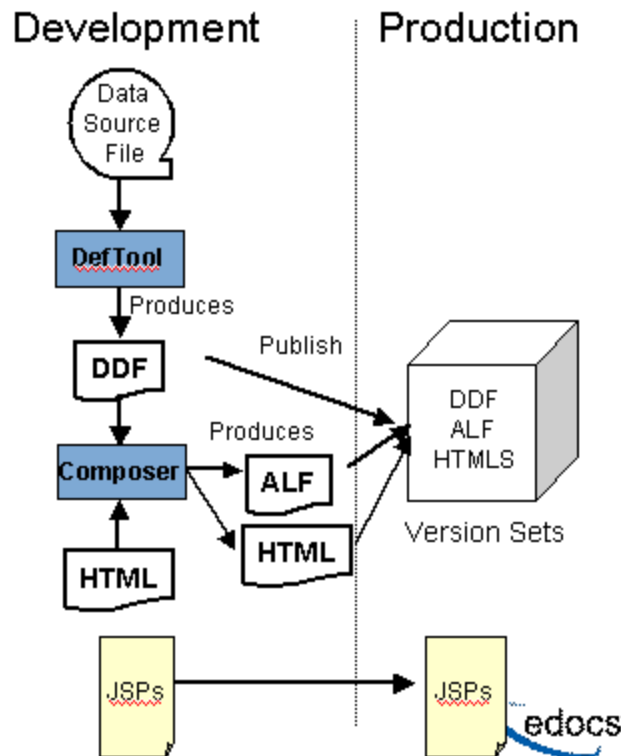
View functionality

- Views provide the ability to sort, filter, page, and total fields easily
- View types include charts, drill-downs, downloads, and printer friendly
- Views can also be used as front ends to messaging & personal address book style functionality

View development

Design Elements

- Version Sets
 - DDF - Provides data extraction rules
 - ALF - Provides formatting and personalization rules
 - HTML with embedded edocs tags - Provides presentation templates



View Types

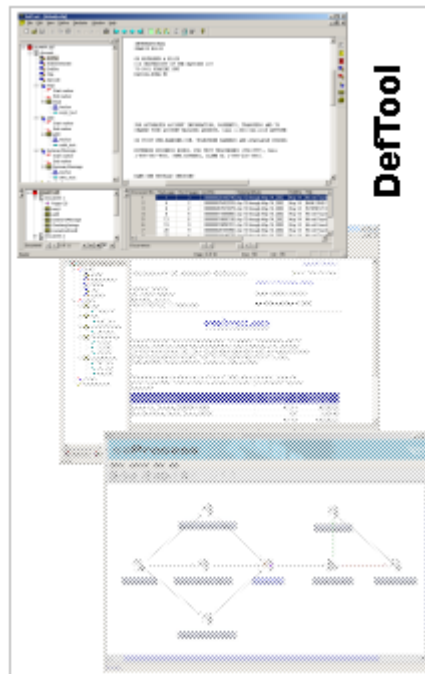
- CSV - Downloadable comma-separated values based on core data
- CHART - Displayable chart generated by Kavachart. Customizable via properties.
- HTML - Standard view type using generated or core data, an ALF and a HTML Template.
- XSLT - Downloadable XML, HTML or anything else generatable by XSLT and XML.
- XML, XML Query - Specialized view run of a detail extractor job to load the statement detail into the database, then generated by an SQL statement off stored data.

DefTool – Data extraction and mapping

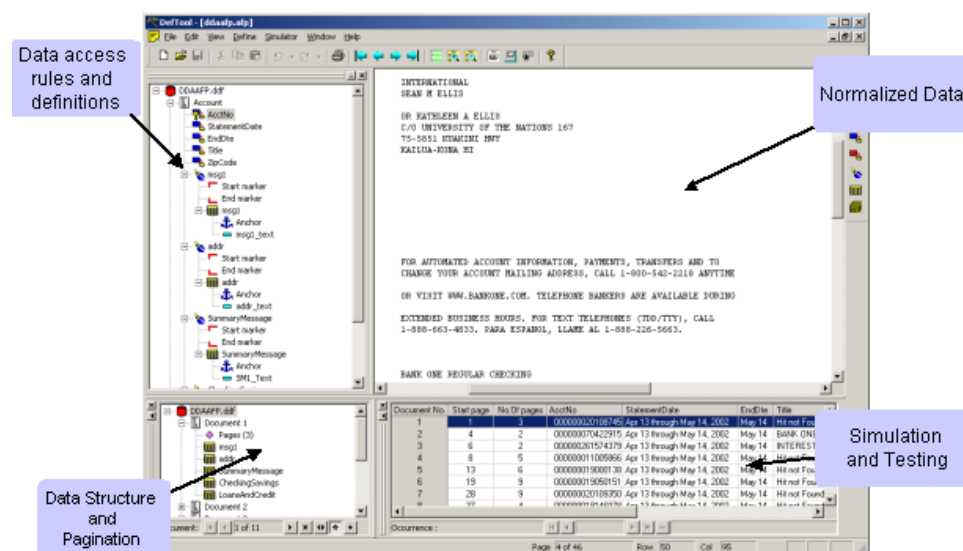
DefTool:

- Is a GUI tool for graphically defining data formats and outputs .DDF files

- Requires no programming or scripting experience
- Supports multi-format data-parsing to allow building connectors into multiple format data sources
- Include powerful post-processing formatting
- Includes a simulator for real-time rules testing



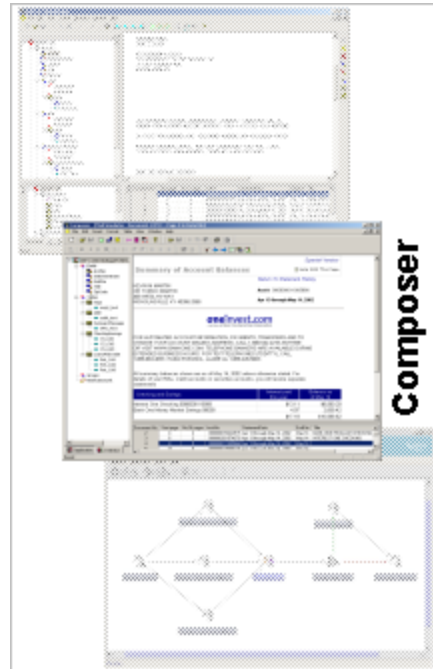
DefTool - Interface



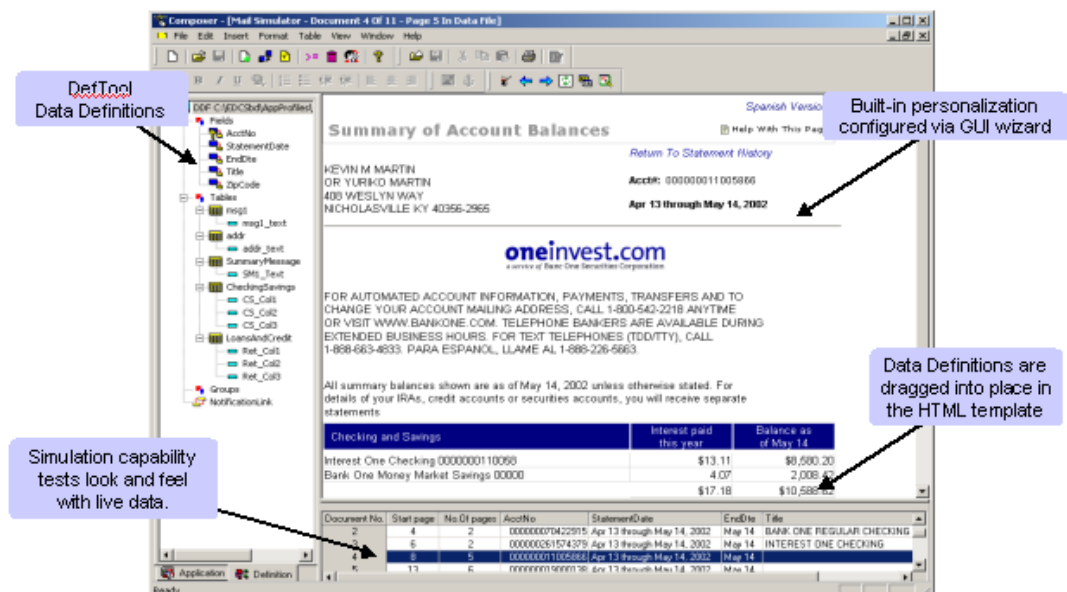
Composer – Business rules and presentation

Composer:

- Is a GUI tool for defining how data and HTML are merged, including WYSIWYG HTML editing
- Requires no programming or scripting experience
- Includes powerful business logic and personalization engine as well as simulator for testing views



Composer Interface



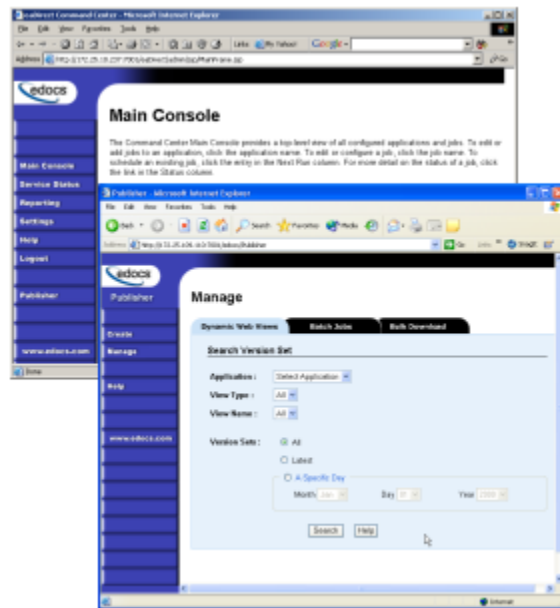
The Command Center

The Command Center provides easy to use, integrated:

- Application configuration
- Run time reporting and statistics
- View and batch process management

Command center includes:

- Main Console - For creating and managing applications and job execution
- Publisher - For managing views



Batch Processes

Telco e-Billing Manager supports standard and custom batch processing.

Batch processes:

- Are known as jobs
- Have one or more steps known as tasks
- Can be scheduled to run periodically

All installed applications have at least one job known as the Indexer Job.

Application: NatlWireless

From this screen, all parameters of the selected job can be modified. To save changes, click the Submit Changes and Schedule button. To Reset the fields, click the Reset button.

Task 1: Scanner

Input File Path: C:\EDCSbd\input\NatlWireless\

Input File Name: *.*

Output File Path: C:\EDCSbd\Data\NatlWireless\

Task 2: Indexer

DDF Path: C:\EDCSbd\AppProfiles\NatlWireless\DDF\

Task 3: IXLoader

Load Method: Direct

Task 4: AutoIndexVolAccept

Action on Index Volume: Intercept to Verify

TBM comes with two types of batch jobs:

- The Indexer job, which:
 - Scans for and processes input billing statements
- Specialized jobs that:
 - Generate email notification
 - Extract specialized details for charts and reports
 - Generate HTML output
 - Generate XML output

Such specialized jobs are often run in preparation of large reports, etc.

Common Jobs

- **Indexer** - Standard job to prepare statement data for presentment. Required for dynamic presentment
- **Email Notification** - Standard job to generates an email notification when an online statement is complete and ready to review.
- **Purge App** - System maintainance job that removes index, email, reporting, detail, and other data from the database
- **Purge Logs** - System maintainance job that removes historical log information.
- **HTML & XML Output** - Statement presentment job which creates either HTML or XML output for each extracted primary key. Not required if live data retrieval is used.
- **Detail Extractor** - Specialized job to generate uploadable XML based on XSLT style sheets.

Creating Applications

TBM Applications are combinations of:

- Application name
- Data source
- Indexer job

Creating an application is a multi-step process composed of:

- Deploy the TBM J2EE application ear
- Create the TBM application
- Publish the indexer files
 - Create the indexer job

Creating applications, adding jobs

See the *TBM Administration Guide* for information on setting up applications and jobs.

Publishing

Publishing is the process of uploading design time artifacts to the edocs system such that applications can use them. You can publish individual sets of files or entire groups of files (bulk).

Version sets are:

- Dated combinations of files
- Are applied based on the current data

Live retrieval of historical bills & statement is handled by:

- Extract data from statement
- Compare against version set dates
- Select version set and apply

Batch Processing Vs. Dynamic Version Set

Batch processing:

- Uses most recent version set
- Applies to: Indexing, Email Notification, HTML Output, XML Output, Detail Extractor

Web view version sets are

- Determined based on data of presentation data
- Apply to: HTML, XML, CSV, XSLT, XMLQuery

Scheduler

The scheduler:

- Must be running to use scheduling & jobs
- Is found in: %EDX_HOME%\estatement\bin
- Is started with: wl_scheduler OR ws_scheduler –start –url t3://<server>:<port>

This chapter describes:

- Core features of the edocs struts & tiles extensions
- TBM application development architecture
- Extending TBM applications to include new menus and pages
- The TBMTags tag library used by TBM, and edocs libraries
- How to configure TBM applications, including resources, properties and internationalization
- How to modify the application look and feel

2.1 Introducing TBM Application Basics

This section describes:

- TBM application fundamentals
- The TBM directory architecture & structure
- Standard pageflows

2.1.1 TBM Application Fundamentals

Telco e-Billing Manager applications:

- Are packaged as an Enterprise Application Archives (.EARs)
- Contain a set of supporting EJBs
- Contain one or more Web Applications (.WARs)
- Must have an associated a Data Definition Name (DDN) created and configured using the Command Center
- Are configured by `app-config.properties`

TBM-based applications:

- Are J2EE applications
- Are based on Struts and Tiles
- Support a set of core functionality including:
 - Login/logout - Authenticate a user
 - Statement presentation - Present summary and statement details
 - Payment - Make payments
 - Profile management - Manage uid/pwd, personal address book etc.
 - Hierarchy - Present and manage company and business based bill presentation hierarchy

2.1.2 Architecture & Structure

TBM application architecture features:

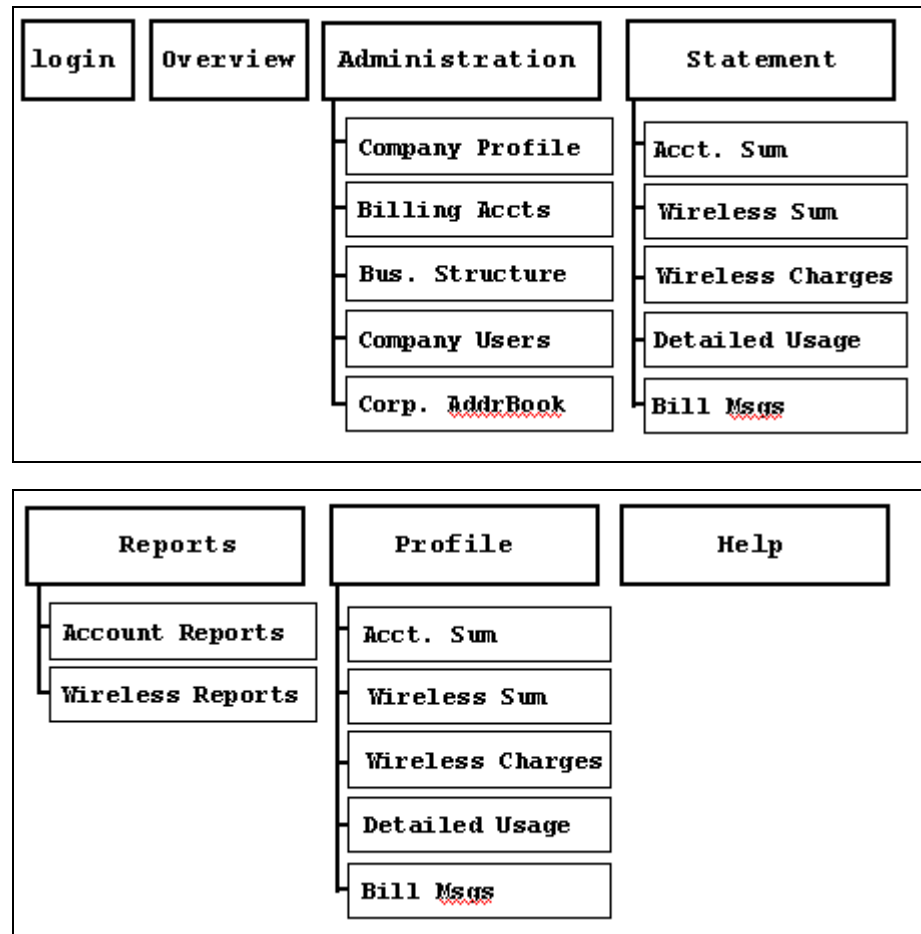
- Clear separation between UI, business logic and controller
 - Struts MVC framework developed by the Apache Software Foundation
 - Separation of roles in a project
- Customer-ready, flexible UI's
 - Use of the Tiles templating engine
 - Internationalization and multi-language support
- Best Practices
 - Standardized validation and error handling
 - No Java (business logic) code in JSP

Application structure

tbm-b2b	Root
lib	Support jars for TBM as well as TBM Application .war
META-INF	Application.xml, defining web application as thmb..
tbm-b2b.war	tbm web application root
_assets	Contains all application assets such as icons, style sheets etc.
_includes	.jsp include files used by the tiles framework
_templates	.jsp templates used by the tiles framework
administration	.jsp specific to company administration
enroll	.jsp specific enrollment support
hierarchy	.jsp specific to hierarchy support
images	immutable image files
login	.jsp specific to login/log out support
META-INF	
profile	.jsp specific to profile support support
WEB-INF	Struts, Tiles, application specific libraries and classes

Note: Every tab normally has a set of JSP

2.1.3 Standard Page Flows

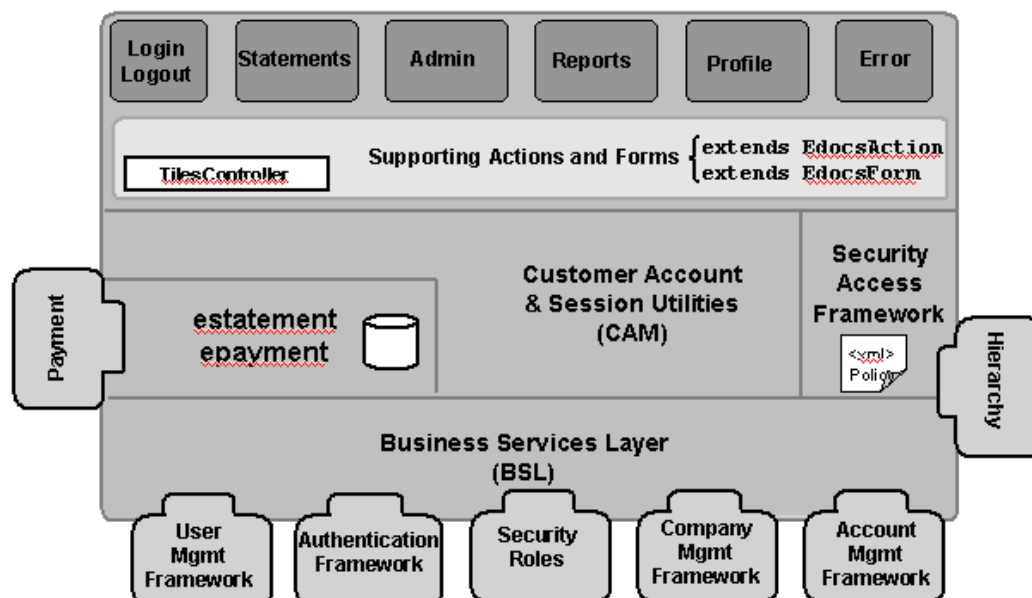


2.2 Struts and Tiles and the edocs View Architecture

This section describes:

- Out of the box architecture components
- Modifying Existing Views
- TBM Tags
- Configuring TBM Applications

2.2.1 TBM Application Architecture



- UserManagement: IUser, IUserProfile, IUserProfileManager
- Security Roles: ISecurityProfile, ISecurityProfileManager
- Authentication: IauthenticationManager
- Company Management Framework: ICompany, ICompanyProfile, ICompanyManager, ICompanyManagerProfile
- Account Management: IBillingAccount, IBilling

2.2.2 TBM Struts and Tiles

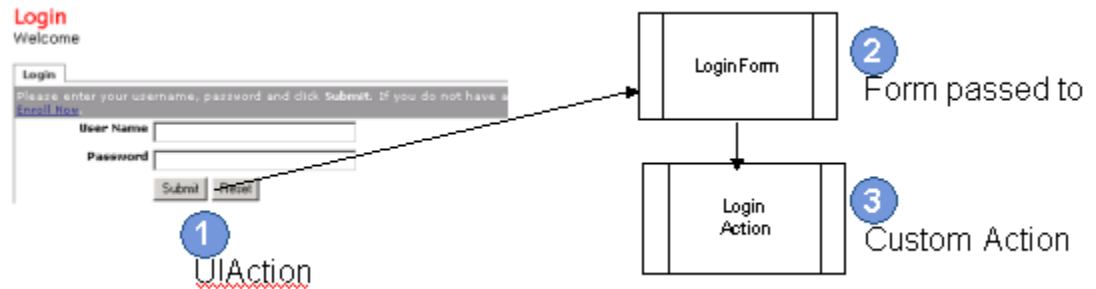
TBM extends the Struts/Tiles framework to:

- Provide specialized struts support infrastructure
- Define core page flow and navigation common to all TBM applications
- Define core page layout, used by navigation, common to all applications
- Provide internationalization and look & feel support

Overall the TBM Struts/files framework provides an extensible environment that can be easily extended to meet customer needs

Typical flow

Pages can include forms and invoke edocs defined actions.



Configuration example

Form:

```
<form-bean name="loginForm"
           type="com.edocs.application.tbm.core.forms.LoginForm"/>
```

Action:

```
<action path="/login/perform"
        type="com.edocs.application.tbm.actions.LoginAction"
        name="loginForm"
        scope="request"
        validate="true"
        input="login">
  <forward name="success"
    path="/hierarchy/main/dashboard/show.do"
    redirect="true"/>
</action>
```

Typical app-config.properties

app-config.properties Example:

```
FILE_PROP_VERSION=TIME_STAMP
```

```
APP_DD_N=TBM your application name here
```

```
# DDN
```

```
DDN.Summary = TBM
```

```
DDN.TBM=TBM
```

```
DDN.Unbilled=Unbilled
```

```
# Views
```

```
DDN_NAME_AccountList = Account_Summary
```

```
#LDE_MAX_CACHE_SIZE=52428800
```

```
...
```

```
%EDX_HOME%/tbm/config/app-config.properties
```

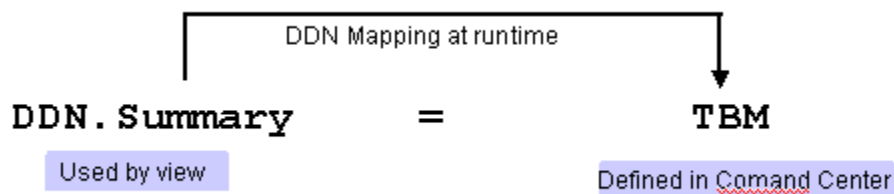
Name of application
in Command Center

DDNs
Summary typically
matches APP_DD_N

Other properties
as required

2.2.3 Mapping DDNs

- *Actual* DDNs represent available sources of data
- Views specify data via *abstract* DDNs
- Abstract DDNs are mapped at run time to actual DDNs via configuration entries.



DDN mapping is important because it allows the developer and system administrator to quickly and easily associate content with data without application coding changes. Applications normally have a default DDN, which also is specified by the edocs application name. The default DDN is used in a variety of places within the struts/tiles hierarchy as well as other locations and should rename TBN. Under normal circumstances the Summary view is mapped to the application DDN although it can be changed to map to any appropriate DDN associated with an indexer job.

You can create other DDNs that map to whatever data is required. Typically a detail extractor job is run, associated with a details job, for the purpose of quick access to account level billing details. Other DDNs typically used are unbilled data, or custom charting or reporting.

2.2.4 Struts Extensions

TBM extends Struts in several areas

- EdocsActionServlet extends ActionServlet to handle TBM session initialization
- EdocsAction extends Action to:
 - Check and force login as required
 - Check for and handle multiple submits
 - Forward to developer implemented doAction
- EdocsActionForm extends ActionForm to handle reset and auto-population of fields

2.2.5 TBM Tiles

TBM uses a tiles structure to define the look and feel of applications. The default structure includes a set of template tiles definitions that define the main functional areas of applications.

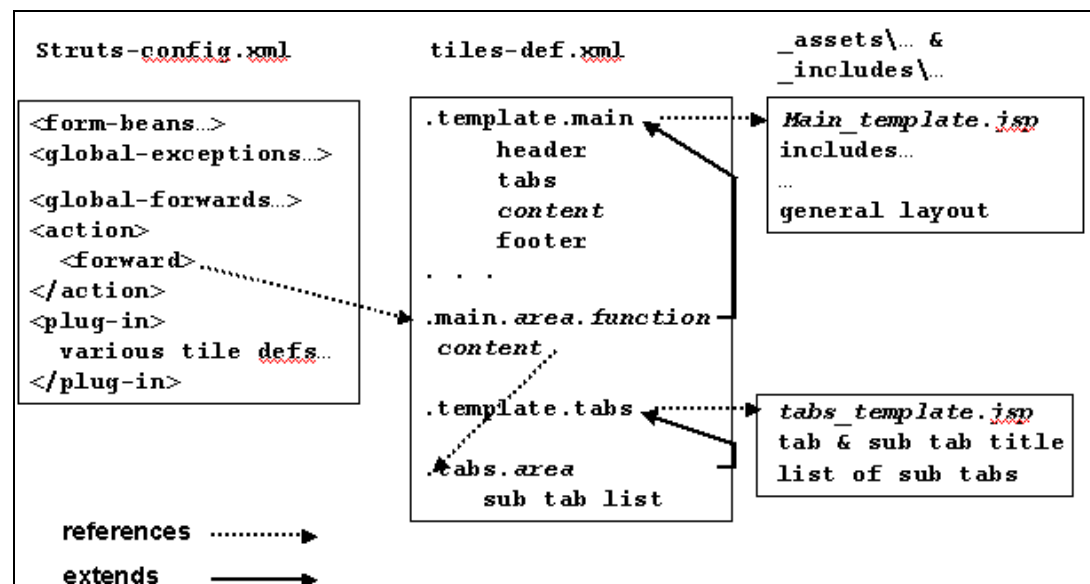
- TBM uses a set of core tiles definitions to define the general TBM structure and actions.

- Tiles *definitions* are used to create templates for each UI functional area.
- Template are further specified by tab and sub tabs.

Some of the TBM template tiles definitions included are:

- .template.main – Defines the overall structure of a presented page. Header, footer etc sections are defined here. tab menus etc.
- .template.tabs - Defines a core tab layout for applications. Pages normally have a set of tabs and sub tabs. The template tabs definition defines the page and other attributes of this list such as selected tab and sub tab list.
- .template.module – Defines a number of top level pages which do NOT include content. Such pages include, login, error, etc.
- .view.setup – Defines the default attributes for a presentable view (core defaults). Common attributes of a view include its name, type and ddn.

General tiles structure



.template.main**.template.main snippet Example:**

based on main template

```

<definition name=".template.main" path="/_templates/main_template.jsp">
  <put name="title" value="pageTitle" type="string"/>
  . . .
  <put name="tabs" value=".template.tabs" type="definition"/>
    <put name="tab" value="Account Summary"/>
    <put name="subtab" value="" type="string"/>
  . . .
  <put name="action" value="/_includes/blank_page.jsp" type="page"/>
  <put name="leftSideNav" value="/_includes/blank_page.jsp" type="page"/>
  <put name="summary" value="/_includes/blank_page.jsp" type="page"/>
  <put name="module1" value="/_includes/blank_page.jsp" type="page"/> . . .
  <put name="module7" value="/_includes/blank_page.jsp" type="page"/>
  <put name="view" value="/_includes/blank_page.jsp" type="page"/>
  <put name="sidebar" value="sidebar.simple" type="definition"/>
  <put name="footer" value="/_includes/footer.jsp" type="page"/>
</definition>

```

includes various navigation tiles

various content tiles

ear-thm-xxx.ear/war-thm-xxx.war/WEB-INF/tiles-def.xml

main.statement.account.summary**Example:**

includes various navigation tiles

```

<definition name=".main.statement.account.summary" extends=".template.main">
  <put name="sectionHeader" value=".template.sectionHeader" type="definition"/>
  <put name="sectionHeadLine" value="Statements"/>
  <put name="subSectionHeadLine" value="Account Summary"/>
  <put name="selectDisplayMenu" value="/_includes/select_display_type.jsp"/>
  <put name="page_name" value="com.edocs.page.name.bill.summary"/>
  <put name="page_description"
    value="com.edocs.page.description.bill.summary"/>
  <put name="tabs" value=".tabs.statements" type="definition"/>
    <put name="tab" value="Statements"/>
    <put name="subtab" value="Account Summary"/>
  <put name="action" value="/_includes/action.jsp" type="page"/>
  <put name="leftSideNav" value="/_includes/side_navbar.jsp" type="page"/>
  <put name="module1" value=".statement.account.summary.view"
    type="definition"/>
</definition>

```

ear-thm-xxx.ear/war-thm-xxx.war/WEB-INF/tiles-def.xml

The statement account summary definition defines a tabbed page which presents an account summary under the statements general tab.

If we examine the definition itself we see that account summary:

- Inherits a section header which has name Statements/Account Summary via the section and subSection head line tiles
- Includes a navigation panel, shown on the left side of the page, which supports account and statement drop down.
- Sets page name and description from the properties com.edocs.page.description.bill.summary and name.bill.summary

- Includes a set of tabs from the .tabs.statements tab definition
- Includes additional navigation via the action and leftSideNav elements
- Overrides the module1 inherited tile to include a view

.template.tabs

.template.tabs:

```
<definition name=".template.tabs" path="/_templates/tab_template.jsp">
  <putList name="tablist">
    <item value="Overview" link="/hierarchy/main/dashboard/show.do"
      classtype="com.edocs...EdocsBaseMenuItem"/>
    <item value="Administration" . . . />
    <item value="Help" link="/view/todo.do"
      classtype="com.edocs...EdocsBaseMenuItem"/>
  </putList>
  <putList name="subtablist"/>
</definition>
```

ear-tbm-xxx.ear/war-tbm-xxx.war/WEB-INF/tiles-def.xml

.tabs.statements

Example:

includes various navigation tiles

```
<definition name=".tabs.statements" extends=".template.tabs">
  <putList name="subtablist">
    <item value="Account Summary"
      link="/view/account_summary.do"
      classtype="com.edocs...EdocsBaseMenuItem"/>
    <item value="Wireless No. Summary"
      link="/view/wireless_number_summary.do"
      classtype="com.edocs...EdocsBaseMenuItem"/>
    ...
  </putList>
</definition>
```

various content tiles

ear-tbm-xxx.ear/war-tbm-xxx.war/WEB-INF/tiles-def.xml

2.2.6 View Tiles

The .view.setup tiles definition defines the default values for view tiles. At a minimum tiles inheriting from view setup set the viewName tile to the view that should be displayed. Additionally you may specify the view type and ddin. Note that the DDN is a virtual or abstract DDN which is mapped by the app-config.properties file to the actual DDN at run time.

.view.setup Example:

```
<definition name=".view.setup" path="/_templates/display_view.jsp"
controllerClass="com.edocs.application.tbm.displayview.DisplayTiledView">
  <put name="viewName" value="DefaultView" />
  <put name="viewType" value="HTML" />
  <put name="ddn" value="TBM" />
</definition>
```

Default view values

.statement.account.summary.view Example:

```
<definition name=".statement.account.summary.view" extends=".view.setup">
  <put name="viewName" value="B2B_AcctSum" />
</definition>
```

Override the view name

ear-tbm-xxx.ear/war-tbm-xxx.war/WEB-INF/tiles-def.xml

View JSP:

```
<%@ include file="/_includes/taglibraries.jsp" %>
<tiles:useAttribute id="viewName"
  name="viewName"
  classname="java.lang.String" />

<tbmtags:getView name="<%=viewName%>" />
```

Obtain view name from
tiles definition

.../war-tbm-xxx.war/_templates/display_view.jsp

Statement Tab examples

The Statement Tab, and all its sub tabs, extend the `.template.main` tile definition.

Each tab defines:

- Display content(titles...)
- Sub tabs
- Navigation
- One or more modules and views

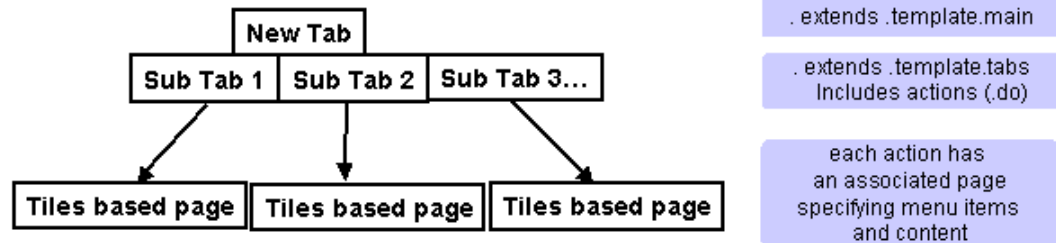
2.2.7 Creating Custom Tabs

You can extend TBM using custom tabs. Custom tabs include:

- Tab and sub tab page definitions
- Struts actions defining page flows
- Various properties files containing localizable text

To create custom tabs:

1. Define the tab structure and flows.
2. Create a tab, sub tabs and tab menu elements
3. Create action mapping.

Tab structure and flows**To create a Tabs & Subtab set:**

1. Add the new top menu tab element to .template.tabs
2. Define a set of sub tabs extending .template.tabs
3. Define page definition(s) that extends .template.main
4. Add properties to Application.Properties for page name and description

Add .template.tabs item

1. Define tabs using the .template.tabs tile.
2. Create sub tabs lists by extending this tile and adding a sub tab list

```

.template.tabs:
<definition name=".template.tabs" path="/_templates/tab_template.jsp">
  <putList name="tablist">
    <item value="Overview" link="/hierarchy/main/dashboard/show.do"
      classtype="com.edocs. .EdocsBaseMenuItem" />
    <item value="Administration" . . . />
    <item value="MyTab" link="/MyTab/action.do"
      classtype="com.edocs.application.thm.core.menus.EdocsBaseMenuItem" />
  </putList>
  <putList name="subtablist" />
</definition>
  
```

Add new tab element
and associated action

ear-thm-xxx.ear/war-thm-xxx.war/WEB-INF/tiles-def.xml

Define Tab definition

Define Sub Tab List:

```

<definition name=".tabs.test" extends=".template.tabs">
  <putList name="subtabList">
    <item value="Sub Tab 1"
          link="action"
          classtype="com.EdocsBaseMenuItem"/>

    <!--2nd and subsequent items... -->
    <item value="Sub Tab 2"
          link="/test/subtab2.do"
          classtype=""/>

  </putList>
</definition>

```

tab sets extend .template.tabs

must include a subtabList

contain a list of sub tab elements

Sub tabs contain:
Tab Display Text
Action
definition of base item class

Sub tab lists contain the list of Name to action mappings displayed for a tab. Each element in a sub tab list consists of three things:

- Value - Containing the sub tab display text
- Link - The action to perform when the sub tab is selected
- classtype - Either com.edocs.application.tbm.core.menus.EdocsBaseMenuItem or a class that extends EdocsBaseMenu item.

Extending EdocsBaseMenuItem

You can extend EdocsBaseMenuItem to support custom visible/invisibility code for a menu element.

EdocsBaseMenuItem extension:

```

package com.mypackagestructure.menus;
import org.apache.struts.tiles.beans.SimpleMenuItem;
import javax.servlet.http.HttpServletRequest;
import com.edocs.application.tbm.core.menus.EdocsBaseMenuItem;
public class MyMenuItem extends EdocsBaseMenuItem {
    public MyMenuItem () { }
    public boolean displayItem(HttpServletRequest request) {
        boolean visible=//sometest
        return visible;
    }
}

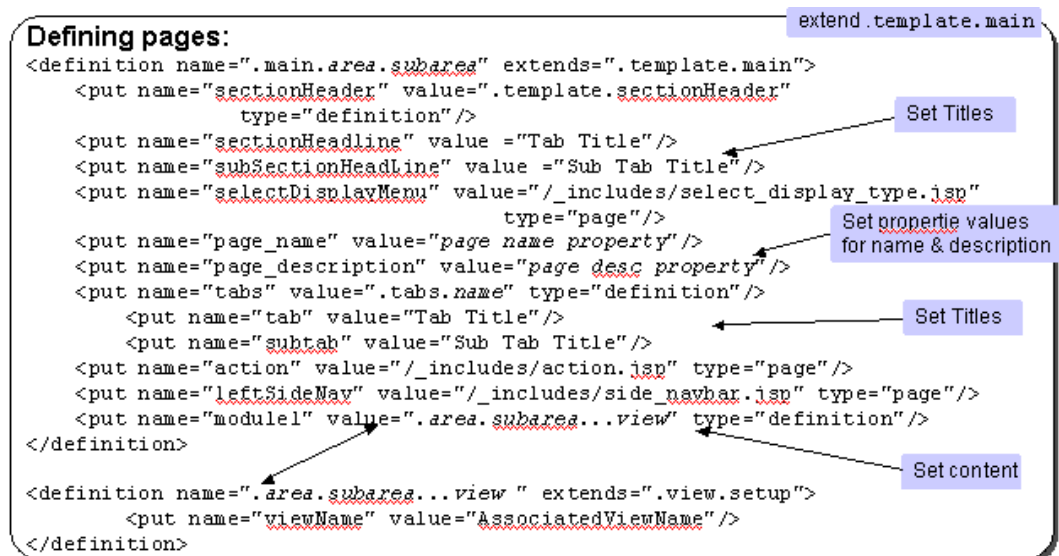
```

Perform custom visibility test

Define Tab Pages

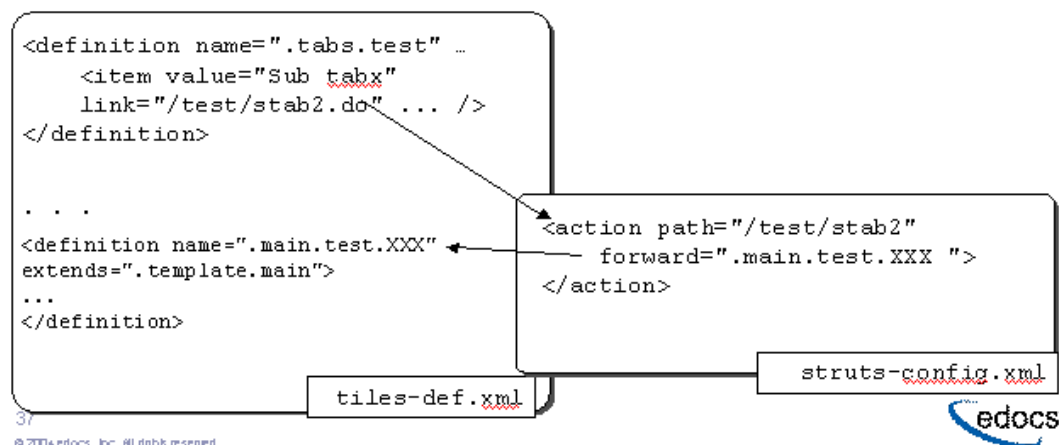
Every tab page extends `.template.main` and specifies a number of elements. The tab titles (main tab and sub tab) defined by both `sectionHeadline` and `subSectionHeadLine`. The tabs display itself uses the tab and subtab tiles to define the actual presentation of the tabs. Two page properties, `page_description` and `page_name` are rendered based on potentially localized version of properties found in `Application.Properties` or a local specific version.

The `module1` tile defines the actual page content. Modules 1-7 are defined and extensible from `.template.main`. Normally module definitions specify another tiles definition containing a view show below the actual `.template.main` example show.



Create Action Mapping

Create Action Mappings in `struts-config.xml`. For each sub tab in the tabs list create an action:



2.3 Development Utilities

TBM includes support for:

- Custom TBM tags
- Custom configuration files
- Custom log4j

Developers can use either custom properties files, or layer properties within the packaged configuration files

TBM includes all required jars and configuration for log4j support.

2.3.1 TBM Tags

TBM comes with a set of custom tags which can:

- Force login or return login/user or account data
- Display a view, view data, or control view display
- Render content in a JSP page (buttons and other content) based on security settings

TBM Tags are specified in JSP include but you can specify them directly.

TBM Tags Tag Library:

```
<%@ include file="/_includes/taglibraries.jsp" %> ← Specified in
or
<%@ taglib uri="/WEB-INF/tbmtags.tld" prefix="tbmtags" %> ↑
```

getView Tag

The `getView` tag displays a named view or the summary view by default.

Syntax:

```
<tbmtags:getView [name="viewName"]/>
```

Example:

```
<%@ include file="/_includes/taglibraries.jsp" %>
<tiles:useAttribute id="viewName" name="viewName"
                    classname="java.lang.String" />

<tbmtags:getView name="<%=viewName%"/>
```

Typically a tiles-def defined page specifies a view by name

summaryDates Tag

The `summaryDates` tag outputs the list of summary dates as a section name. It returns a SELECT= statement populated with the current set of selectable summary dates.

Syntax:

```
<tbmtags:summaryDates property="fieldname in form"
                        [styleclass="optional style class"]
                        [onchange="optional on change method"]
                        [sortorder="optinal sort order=asc|desc|none"]/>
```

Example:

```
<%@ include file="/_includes/taglibraries.jsp" %>

<html:form action="myaction.do">
  <tbmtags:summaryDates property="date"
                        styleclass="formlabel" sortorder="desc"/>
</html:form>
```

summaryInfo Tag

The `summaryInfo` tag returns any summary field based on the currently selected statement. Common summary fields are:

- AcctNum
- CurrentCharges
- PymtType
- StatementDate
- AmountDue
- LateFee

Syntax:

```
<tbmtags:summaryInfo name="summary field"/>
```

Example:

```
<%@ include file="/_includes/taglibraries.jsp" %>

The amount due for Account <tbmtags:summaryInfo name="AcctNum"/> is
<tbmtags:summaryInfo name="AmountDue"/> for statement
<tbmtags:summaryInfo name="StatementDate"/>.
```

checkLogin Tab

The `checkLogin` tag forces the application to the login page if no valid user is in the session. Use this tag on any page requiring a login.

Example Check Login: If user

```
<%@ include file="/_includes/taglibraries.jsp" %> ← Specified in
or
<%@ taglib uri="/WEB-INF/tbmtags.tld" prefix="tbmtags" %> ↑
<tbmtags:checkLogin/>
```

GetGeneralAccountInfo

getGeneralAccountInfo returns a specified account or billing account level property

Example Check Login: If user

```
<%@ include file="/_includes/taglibraries.jsp" %>
or
<%@ taglib uri="/WEB-INF/tbmtags.tld" prefix="tbmtags" %>

<tbmtags:getGeneralAccountInfo
    [billingAccount="billing account"]

    property="property to retrieve"

    [accountLevelRequest=true|false]/>
```

Optional Billing Account Number
Assumed to be a billing request if set
and all information will be
from the billing account given

Property to return

Assumed billing request
if set. Defaults to false

Here are some common Properties used by getGeneralAccountInfo. See the IAccount documentation for a complete list.

- UserId - Currently logged in user
- Associated Email Address - EmailAddress";
- First name - FirstName
- Last Name - LastName
- Default – DDN
- Associated Account Number

2.3.2 Working With Properties

TBM applications can access properties via Configuration objects.

Configuration objects:

- Can load and store string, integer and sets of configuration values
- Auto update based on changes in backing store
- Auto store based on changes to in memory instances

Configuration Example:

```
app-config.properties:
usesConfig.myProperty=value
userConfig.timeout=1000
userConfig.list=value1,value2,value3
```

Java Class:

```
import com.edocs.application.tbm.config.Configuration;
Configuration c = Configuration.getConfiguration();
String result = c.getValue("usesConfig.myProperty ");

Int timeout = c.getIntValue("userConfig.timeout",10);

String[] listof = c.getValues("userConfig.list");

c.setValue(String key, String value);
```

May specify a properties file name on the system CLASSPATH

Returns value or default

Returns array representing values

Persist a value

2.3.3 Working With Logs

TBM Applications can perform custom logging via log4j. Log4j supports:

- Controlling logging levels at runtime
- Configuring custom loggers
- Defining log message format

Classes can instantiate loggers, write logging code and then control log output via configuration files.

Log Example:

```
config\log4j.xml
```

```
<category name="com.myco">
  <priority value="info|debug|warn|fatal|error">
</category>
```

Select msg level to log for this category

Java Class:

```
package com.myco;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
public class usesLogging {
private Log m_log = LogFactory.getLog(usesLogging.class);
  public void logsMsg() {
    if ( m_log.isDebugEnabled())
      m_log.debug("in logsMsg");
  }
}
```

Create a logger & log to it!

2.3.4 Developing Custom Actions

TBM custom struts actions:

- Extend EdocsAction
- May use forms which extend EdocsActionForm
- Can access environment using the CAM/SessionUtils layer.

2.3.5 Customer Account Manager

See the CAM javadocs delivered in this SDK for implementation details about the Customer Account Manager (CAM) components of TBM.

The Customer Account Manager (CAM):

- Provides information required to manage that state of a customer session.
- Includes a hierarchy of objects including customer, customer accounts, billing information etc.
- Uses a factory metaphore to access underlying customer data.

The Customer Account Manager, or CAM, is a subsystem designed for the purpose of managing access to underlying data. In the context of an action class CAM is typically used to access customer state information. The CAMClassFactory is a singleton object which accesses both the current session and the underlying BSL subsystem to return requested data.

Customer account numbers are accessed via the ICustomerAccount interface which exposes a getAccountNumber method.

```
Accessing a Customers from within an action
import com.edocs.application.tbm.cam.api.*;
...
ICustomer cust = CAMClassFactory.getCustomer(httpServletRequest);
ICustomerAccount acct = cust.getCurrentAccount();
String acctNumber = acct.getAccountNumber();
```

Obtain the current customer

Use ICustomer to obtain accounts

Common CAM interfaces

Interface	Description and Use
CAMClassFactory	Create or access customer information. Common use: <code>ICustomer getCustomer(HttpServletRequest);</code>
ICustomer	Return attributes of Customers. Common use: <code>ICustomerAccount getCurrentAccount();</code> <code>String getAttribute(String name);</code> <code>String getUserID();</code>
ICustomerAccount	Returns attributes of customers such as account #'s and summary data Common use: <code>String getAccountNumber();</code> <code>IAccountSummary getSummaryInfoDefaultDDN();</code>
IAccountSummary	Returns or sets data about the associated summary such as valid dates, current date, amount due, and parameters Common use: <code>String[] getStatementDates();</code> <code>String[] getAmountDue();</code>

CAM example

Obtain the range of summary dates for customers current account:

```
import com.edocs.application.tbm.cam.api.CAMClassFactory;
import com.edocs.application.tbm.cam.api.IAccountSummary;
import com.edocs.application.tbm.cam.api.ICustomer

...

ICustomer cust = CAMClassFactory.getCustomer(request);

ICustomerAccount acct = cust.getCurrentAccount();

IAccountSummary summary = acct.getSummaryInfoDefaultDDN();

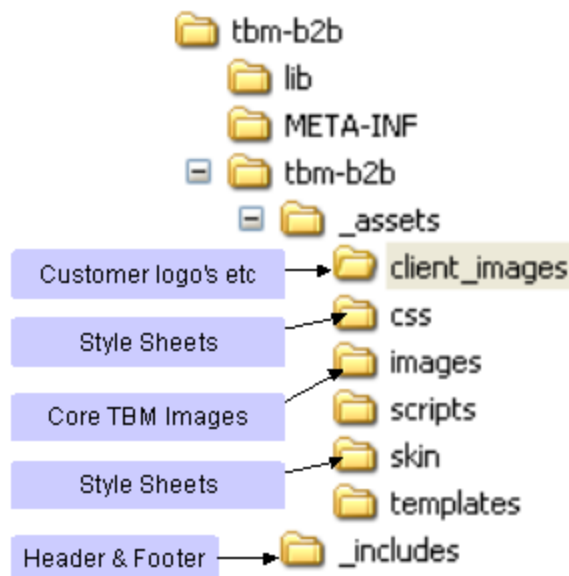
String summaryDates[] = summary.getStatementDates();
```

2.4 Look and Feel

To manage application look and feel you can:

- Manage template files such as header/footer.jsp
- Set a core color and text scheme using style sheets
- Manage images

Place these items in the TBM directory as follows:



Define application header & footer content using:

tbm-[b2b|b2c_assets_includes\header.jsp & footer.jsp

```

<%@ include file="/_includes/taglibraries.jsp" %>
<div id="navtop">
  <div id="navtopblackbar">&nbsp;</div>
  <div id="navtoplogobar">
    
    <div id="navtoplinks">
      <a href="#">Contact Us</a> |
      <a href="<%= request.getContextPath()
        %>/logout/perform.do">Logout</a>
    </div>
  </div>
  <div id="navtopredbar">&nbsp;</div>
</div>

```

2.4.1 Message Rendering and Localization

UI text content is obtained extensively using resource properties.

Resource properties are:

- Read from WEB-INF\classes\application[_locale].properties based on run time locale
- Rendered by <bean:message key="property name"> tags

You can localize text resources by:

- Creating a locale specific application properties file
- Adding or replacing property text with localized versions

2.4.2 Style Sheets

TBM Application style is controlled by style sheets found in:

tbm-[b2b|b2c_assets\css\ and tbm-[b2b|b2c\skins\

_assets\skins\skin.css

- Primary CSS for look and feel
- Define the look and feel of pages and navigation
- Can change definitions (fonts, size etc)
- Should NEVER add or remove definitions

assets\css\edx.css

- Support of additional look and feel
- Exists to be customized
- Can contain new additions

_assets\css\hierarchy.css

- Primary CSS for look and feel for hierarchy
- Can change definitions (fonts, size etc)
- Should NEVER add or remove definitions

2.4.3 Images

Images and logos exist to further define application look and feel

_assets\client_images\

- Contains replaceable images
- Should contain any new images

skins\

- Contains UI images for tabs etc
- Can be replaced, but must maintain size

images\

- Contains core images which can be modified but no new images added
- Can be replaced, but must maintain size

3

Statementing and Content Access

This chapter describes how to extend and customize views, including:

- View architecture basics
- Downloadable view support
- How to write and configure post processors
- How to write and configure live data extraction

3.1 View Architecture Basics

TBM simplifies view presentment via a custom tiles controller, view jsp and view tiles.

View tiles:

- Use TBMTags via `template.jsp` to render view content.
- Defines `viewName`, `viewType`, and DDN tiles, used by the view controller to specify content.

You can write custom template view jsp pages to manage special requirements such as download views

View architecture

TBM Supports two source data types:

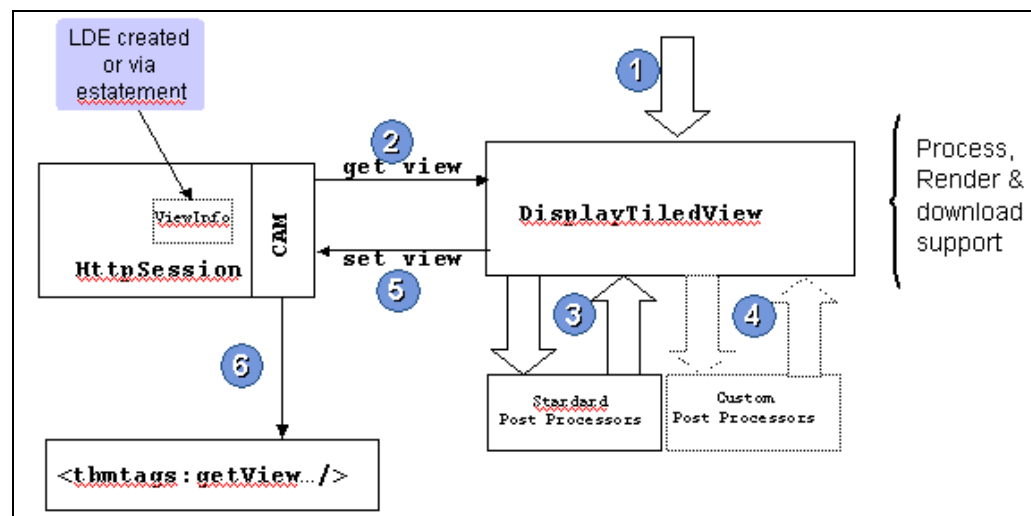
- Indexed views, obtaining data from indexed DDNs and requiring no specialized actions
- Live data views, obtaining data from custom sources

TBM View processing is extended based on a specialized tiles controller `DisplayTiledView`, which:

- Provides view rendering support
- Provides view post processing

- Is extensible, providing support for custom post processors and live data

Display Tiles run time architecture



The Display Tiles Controller hooks run time requests to display views and executes as follows:

1. The struts/tiles framework calls the DisplayTiledView (DTV) controller.
2. The DTV controller then uses the CAM layer to obtain a view stream for processing.
3. Standard post processors and custom post processors are applied in the order defined in the view definition.
4. The resulting, perhaps changed, view is set back into the underlying session via CAM.
5. The actual JSP tied to the view via its `.view.setup` defined definition is run and the `tbmtags:getView` tag renders the view.

3.2 Downloadable View Support

TBM supports creating download views:

1. Create and publish a download view
2. Use or modify download template.jsp

Downloadable JSP Example:

```
<%@ include file="/_includes/taglibraries.jsp" %>
<tiles:useAttribute id="viewName" name="viewName"
    classname="java.lang.String" />
<%
String fileName=(String) request.getAttribute("FILE_NAME");
%>

<tbmtags:getView name="<%=viewName%>" fileName="<%=fileName%>" />

}
```

Extend or modify
as required

Tiles controller supports downloads via filename

_templates/display_download_view.jsp

3. Define downloadable view tile:

Download Tile Template:

```
<definition name=".template.customdownload"
    path="/_templates/display_download_view.jsp"
    controllerClass=
    "com.edocs.application.tbm.displayview.DisplayTiledView">
    <put name="viewName" value="DefaultView" />
    <put name="viewType" value="XSLT" />
    <put name="ddn" value="TBM" />
</definition>
```

If many tiles to be defined
create a custom template

Download Example:

```
<definition name=".download.someView" extends=".template.customdownload">
    <put name="viewName" value="publishedXSLTViewName" />
    [ <put name="ddn" value="TBM" /> ]
</definition>
```

Set View Name
Set DDN if required

4. Use new tile in appropriate page

3.3 Writing and Configuring Post Processors

Post processors are used to transform data in some way after it has been returned from a data source. Views can be associated with one or more Post Processors.

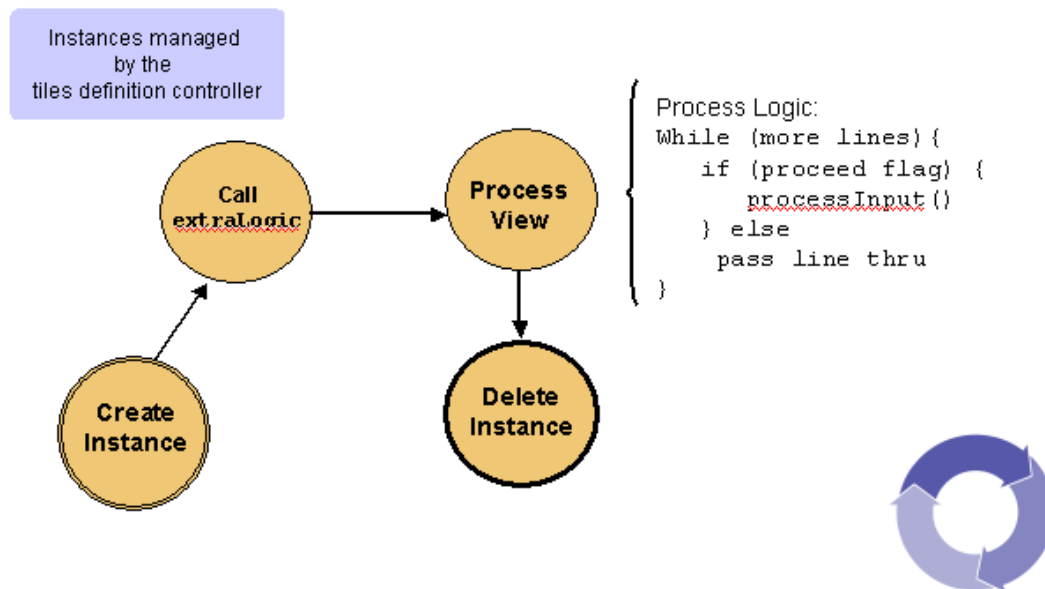
Post Processors:

- Are called after a view has been obtained from a data source
- Can transform data programmatically
- Can be parameterized for a specific view in a tiles definition
- Can be chained one after another

Examples of post processing include:

- Replacing text with a link, for example to insert personal address book data.
- Adding new text, for example replacing a phone number with a name.
- Summarizing data as its presented (usage statistics).
- Logging information about data.

Post processor lifestyle



3.3.1 How to Write a Post Processor

To write a post processor:

1. Write a class that extends PostProcessor (writing two specific methods)
2. Register the postprocessor logical name in app-config.properties
3. Modify the tiles def which specifies the view to be post processed to reference the view logical name
4. Optionally modify the tiles def to supply additional arguments to the view

Each step is described here:

1. Extending PostProcessor

Post Processor Example:

```
import com.edocs.application.tbm.displayview.PostProcessor;
import com.edocs.application.tbm.core.common.ViewInfo;
import javax.servlet.http.HttpServletRequest;
public class MyPP extends PostProcessor {
    public MyPP() {}
    protected void handleExtraLogic(ViewInfo viewInfo,
                                    HttpServletRequest request,
                                    HashMap parameters)
                                    throws Exception {}
    protected String processInput(String inputLine)
                                    throws Exception {}
}
```

Required Imports

Extends

Required Methods

```
import com.edocs.application.tbm.displayview.PostProcessor;
import com.edocs.application.tbm.core.common.ViewInfo;
import javax.servlet.http.HttpServletRequest;
import org.apache.commons.logging.*;
public class MyPP extends PostProcessor{

    private static Log m_cLog = LogFactory.getLog(MyPP.class);
    //-- definitions for initial and final string to match. --
    private static final String cszBeginMatchString = "startpattern";
    private static final String cszBeginMatchTemplate = "XXXXXXXXXXXX";
    private static final String cszEndMatchString = "end pattern";
    private static final String cszEndMatchTemplate = "XXXXXXXXXXXX";

    // -- matching classes. --
    private OutputThroughMatch m_FindBeginMatch = null;
    private RetainThroughMatch m_RetainMatch = null;

    public MyPP() {
        m_FindBeginMatch = new OutputThroughMatch(
                                cszBeginMatchString,
                                cszBeginMatchTemplate );
        m_RetainMatch = new RetainThroughMatch(
                                cszEndMatchString,
                                cszEndMatchTemplate )
    }

    protected void handleExtraLogic(ViewInfo viewInfo,
                                    HttpServletRequest request,
                                    HashMap parameters)
                                    throws Exception {

        if (m_cLog.isDebugEnabled()) {
            String aParam = (String)parameters.get(new String("key"));
            String anotherParam = (String)parameters.get(
                                                new String("key2"));
        }
    }

    protected String processInput(String inputLine) throws Exception {
        if (m_cLog.isDebugEnabled())
            m_cLog.debug("MyPP -- entering with string \""+
                        inputLine + "\"");

        String t_szRet = m_FindBeginMatch.process(inputLine);
        if (m_FindBeginMatch.isMatchComplete()) {
            // found the beginning, look for the end
            if (m_RetainMatch.isMatchComplete())
                setProceed(false);
        }
        return t_szRet;
    } // -- end of processInput
}
```

2. Register the Post Processor

To register the Post Processors:

- Register in app-config.properties
- Give it a logical name used by tiles definitions

Registering a Post Processor:

In general:

```
PostProcessors=Name1[, name2...]  
PostProc.Name1=fully qualified name
```

For example:

```
PostProcessors=MvPP  
PostProc.MvPP=com.educs.edu.MyPostProcessor
```

%EDX_HOME%/tbn/app-config.properties

3. Modify the Tiles Definition

Post Processors are specified to be used with views via an associated tile.

Specifying a Post Processor:

In general:

```
<definition name=".some.definition" extends=".view.setup">  
  <put name="viewName" value="Name of View"/>  
  ... other parameters  
  <put name="postProcessor" value="MvPP"/>  
</definition>
```

Must match name in
app-config.properties!

For example:

```
<definition name=".analysis.expensive.tx.chart" extends=".view.setup">  
  <put name="viewName" value="Report_ExpensiveTxCH"/>  
  <put name="viewType" value="HTML"/>  
  <put name="ddn" value="Itemisation"/>  
  <put name="subIdx" value="true"/>  
  <put name="postProcessor" value="ReportChart"/>  
</definition>
```

4. Provide Optional Parameters

Parameters as passed to post processors via a viewParam tile.

Specifying a Post Processor:

In general:

```
<definition name=".some.definition" extends=".view.setup">
  <put name="viewName" value="Name of View"/>
  ... other parameters
  <put name="postProcessor" value="MyPP"/>
  <put name="viewParam" value="key=value[|key2=value2...]" />
</definition>
```

For example:

```
<definition name=".analysis.expensive.tx.chart" extends=".view.setup">
  . . .
  <put name="postProcessor" value="ReportChart"/>
  <put name="viewParam" value="start=MyMatch|startMask=XXXXXXX" />
</definition>
```

```
{WEBAPPLICATION}/WEB-INF/tiles-*.xml
```

3.3.2 Pattern Processing

View Processors typically:

- Search for a starting and ending character sequence
- Extract the contents between the starting and ending sequence for use as a data retrieval key
- Replace the sequence

The com.edocs.domain.telco.tagprocessing package contains four classes for pattern processing: OutputThroughMatch, ExactThroughMatch, RetailThroughMatch, DiscardUpToMatch

The heart of view post processing is pattern matching. Typically a post processor processes content by finding a starting and ending sequence, extracting the intermediate content and uses the content in some way. In Personal Address Book applications the intermediate content, normally a phone number, is used as a key into a table and the number itself replaced with a URL which displays address book content.

In support of pattern matching four classes are provided, all extending ProcessThroughMatch.

Classes of the package com.edocs.domain.telco.tagprocessing are:

- OutputThroughMatch - Returns everything up to and including the match string from the input
- ExactThroughMatch - Retains all input in a special internal buffer
- RetailThroughMatch - Retains the input up to the but not including the match
- DiscardUpToMatch - Discards all input to to the match.

OutputThroughMatch

Return the output of a match-up through and including the match string.

OutputThroughMatch:

```
public class OutputThroughMatch
    extends ProcessThroughMatch ... {

    OutputThroughMatch(String match, String mask);

    String process(java.lang.String szInput);

    public boolean isMatchComplete();

    String getExtraData();
}
```

Create an instance

Search for a match

match found?

Return data after the match

Example

Match a give starting string including start:

```
import com.edocs.domain.telco.tagprocessing.*;
OutputThroughMatch start = new OutputThroughMatch("StartsWith*F*",
    "XXXXXXXXXXXXXX");

String result = start.process(inputLine);
if (start.isMatchComplete()) {
    String remaining start.getExtraData();
}
```

Match exactly the text

Match Character	Meaning
'a' or 'A'	Match the character in the provided match string or a whitespace
'w' or 'W'	Match whitespace
'x' or 'X'	Match exactly the character in the match string

3.4 Live Data Extraction

DDNs can be:

- Static - based on an a batch job for content
- Dynamic, or live - based on a data stream managed dynamically by a live data stream and the LDE Manager

Views using dynamic perform identically to traditional views, and can employ all the same mechanisms such as view post processing

Live data processing is managed via a custom data stream and the lde manager. Developers write a custom action class, which interacts with the current session for customer information, and the view management system to access data and present it dynamically.

From the perspective of view presentment, live data streams behave identically to traditional data streams and can take advantage of post processors etc. For all practical purposes dynamic data is no different, to the view subsystem, then previously processed data.

3.4.1 About the LDE Manager

The LDEmanager manages dynamic view content using a least recently used algorithm. Data content returned by data sources is stored and then purged over time. Several configuration paramaters control how data is managed.

The LDE Manager:

- Caches LDE Data streams for presentation, periodically expiring and clearing content
- Interacts with the view subsystem to provide dynamic content for presentation
- Is tunable based on configuration settings in `app-config.properties`

LDE Configuration settings:

```
LDE_MAX_CACHE_SIZE=10485760    default 10MB
LDE_EXPIRATION_PERIOD=600000    default 10 minutes, in milliseconds
LDE_CLEANUP_PERIOD=300000       default 5 minutes, in milliseconds
```

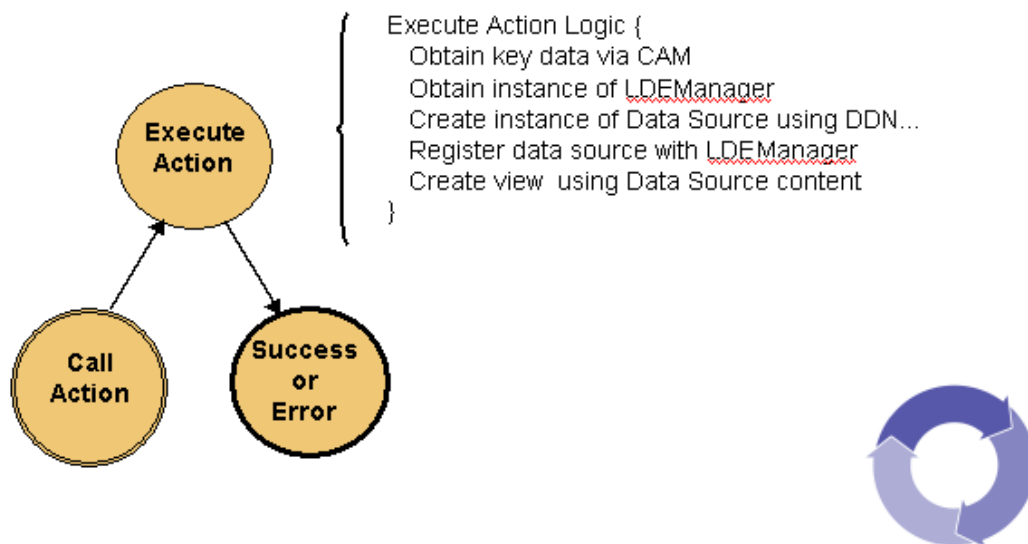
```
%EDX_HOME%/tbn/app-config.properties
```

The LDE Cache size represents the maximum size of view data which can be kept in memory and defaults to 10mb.

The LDE Cleanup period represents the time period used to sweep through the cache and deleted expired objects and defaults to every 5 minutes.

The LDE Expiration period represents the lifetime of a cached object and defaults to 10 minutes.

LDE Process



3.4.2 About LDE Data Streams

LDE Data Streams:

- Behave identically to traditional view data streams and are used by the LDE Manager to access run time data
- Must ultimately implement public interface `com.edocs.domain.telco.lde.ILDEStream`
- Often extend abstract class `com.edocs.domain.telco.lde.ALDEStream` or `com. . .lde.AZipMemoryLDEStream`

ILDEStream Interface:

```

public interface ILDEStream {
    public void reset();
    public InputStream getContent();
    public long getTimestamp();
    public int getSize();
    public String getDDN();
}
    
```

3.4.3 Developing Live Data Extractors

There are five steps to writing Live Data Extractors:

1. Create a representative DDN for the data stream using the LDE data source

2. Write a class that extends `ILDEStream`, `ALDEStream` or `AZipMemoryLDEStream`
3. Write and configure an action class that registers the LDE Stream and forwards to view display page
4. Modify `edx.config.[bat|sh]` to support the LDE protocol
5. Define view definition, extending `.view.setup` defining the view parameters

Each step is described below.

1. Register the DDN

1. Login to the edocs Command Center
For example `http://localhost:7001/edocs`
2. From the main window select Create New Application
3. Enter a suitable name
4. Enter the lde data source
For example: `edx/{appname}/ejb/LdeDataSource`

2. Write the LDE Data Source Class

Writing LDE Data Classes:

1. Write a class that extends:
 - `ILDEStream`
 - `ALDEStream`
 - `AZipMemoryLDEStream`
2. Write a Constructor and store the DDN and optionally obtains configuration information.
3. Write a method, traditionally called `getData`, which:
 1. Obtains a byte array of data
 2. Stores it using the inherited `internalSetContent` method
 3. Returns true or false indicating whether data is loaded.

LDE data source example

```
import java.io.ByteArrayOutputStream;
import com.edocs.domain.telco.lde.AZipMemoryLDEStream;
import com.edocs.application.tbm.config.Configuration;
import com.edocs.domain.telco.lde.LDEManager;
import java.io.*;

public class SimpleLDEDataSource extends AZipMemoryLDEStream {
    /** file to obtain content from */
    private String m_ContentFile = null;
```

```

public SimpleLDEDataSource(String ddn) {
    super(ddn);
    loadConfig();
}

public boolean getData(String key) {
    byte buffer[] = null;
    //-----
    // Get a fileName, read the file and set the content
    // to the byte array
    // -----
    if (m_ContentFile != null && m_ContentFile.length() > 0) {
        File file = new File(m_ContentFile);
        int size = (int)file.length();
        buffer = new byte[size];
        try {
            InputStream fileIn = new FileInputStream(m_ContentFile);
            fileIn.read(buffer);
        } catch (IOException e) { e.printStackTrace(); }
    }

    // call setinternalContent if the byte array is not null
    if (buffer != null && buffer.length > 0) {
        internalSetContent(buffer);
    }
    else
        return false;
    return true;
}

private void loadConfig() {
    Configuration cConfig = Configuration.getConfiguration();
    if (cConfig != null) {
        String value = cConfig.getValue("simplelde.contentFile");
    }
}
}

```

3. Write Action Class

Writing LDE Data Classes:

1. Write a class that extends `EdocsAction`
2. Implement `doAction` which typically:
 - a. Obtains an instance of an LDE Manager
 - b. Obtains the DDN and other key data required by the datasource
 - c. Creates an instance of the underlying data source
 - d. Registers the datasource with the LDE Manager
 - e. Creates and populate one or more views

LDE Action Class: Imports

```
import com.edocs.application.thm.core.common.*;

import com.edocs.application.thm.cam.api.*;

import com.edocs.application.thm.core.common.SessionUtils;

import com.edocs.application.thm.core.common.ViewInfo;

import com.edocs.domain.telco.lde.LDEManager;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.*;
```

Required Constants
And classes for
building edocs actions

User Information via CAM

Session

View Support

LDE Manager

Request & Response

Struts Support

LDE Action Class: `doAction` - Obtain DDN and sub key data

```
...

public class SimpleLDEAction extends EdocsAction {

    public ActionForward doAction(ActionMapping actionMapping,
                                  ActionForm actionForm,
                                  HttpServletRequest req,
                                  HttpServletResponse resp)
        throws RuntimeException {

        ICustomer t_cust = CAMClassFactory.getCustomer(req);
        ICustomerAccount t_acct = t_cust.getCurrentAccount();
        IAccountSummary t_sum = t_acct.getSummaryInfoDefaultDDN();
        // use t_sum for any other sub key data..
        IDDNMap ddnMap = t_cust.getDDNMap();
        String ddn = ddnMap.getDDN("Simple");

        ...
    }
}
```

LDE Action Class: `doAction` - Obtain instance of data source and LDE manager

```

. . .
// data source might take addn args

SimpleLDEDataSource ds = new SimpleLDEDataSource (ddn [,. . .]);

LDEManager ldeManager = LDEManager.getInstance();
if (ldeManager == null )
    return actionMapping.findForward(RIConstants.ERROR);

if (ds.getData() != true)
    return actionMapping.findForward(RIConstants.ERROR);

String docId = ldeManager.setContent(ds);
if (docId!= null && docId.length() > 0) {
    // build view
}

. . .

```

LDE Action Class: `doAction` - build view

```

. . .
if (docId== null || docId.length() == 0)
    return actionMapping.findForward(RIConstants.ERROR);

ViewInfo view = new ViewInfo();
view.setDdn(ddn);
view.setDocId(docId);
view.setViewName("SimpleSummary"); // SimpleDetails
view.setViewType("HTML");
SessionUtils.setView(httpServletRequest, view);

return actionMapping.findForward(RIConstants.SUCCESS);

}
}

```

LDE action class example

```

import com.edocs.application.tbm.core.common.*;
import com.edocs.application.tbm.cam.api.*;
import com.edocs.application.tbm.core.common.ViewInfo;
import com.edocs.domain.telco.lde.LDEManager;
import org.apache.struts.action.*;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.commons.logging.*;

public class SimpleLDEAction extends EdocsAction {
    public ActionForward doAction(ActionMapping actionMapping,
                                ActionForm actionForm,
                                HttpServletRequest req,
                                HttpServletResponse resp)
        throws RIException {

        try {

```

```

ICustomer t_cust = CAMClassFactory.getCustomer(req);
ICustomerAccount t_acct = t_cust.getCurrentAccount();
IAccountSummary t_sum = t_acct.getSummaryInfoDefaultDDN();
IDDNMap ddnMap = t_cust.getDDNMap();
String ddn = ddnMap.getDDN("Simple");
SimpleLDEDataSource ds = new SimpleLDEDataSource (ddn );
LDEManager ldeManager = LDEManager.getInstance();
if ( ldeManager == null )
    return actionMapping.findForward(RIConstants.ERROR);
if (ds.getData() != true)
    return actionMapping.findForward(RIConstants.ERROR);
String DocId = ldeManager.setContent(ds);
if (docId == null || t_szDocId.length() == 0) {
    return actionMapping.findForward(RIConstants.ERROR);
}
ViewInfo view = new ViewInfo();
view.setDdn(szUnbilled_ddn);
view.setDocId(t_szDocId);
view.setViewName("SimpleSummary"); // SimpleDetails
view.setViewType("HTML");
SessionUtils.setView(req, view);

} catch(RISessionNotFoundException re) {
    return
actionMapping.findForward(RIConstants.ERROR);
} catch(java.lang.Exception ex) {
    return actionMapping.findForward(RIConstants.ERROR);
} // end try

return actionMapping.findForward(RIConstants.SUCCESS);
} // end doAction
}

```

4. Support the LDE Protocol

The LDE protocol is specified via the java.protocol.handler.pkgs property.

LDE Protocol support: `edx.config[.bat|.sh]`

By default the LDE protocol is loaded in TBM via a statement in `edx.config` similar to:

```

@set PROTOCOL_HANDERS=com.edocs.protocol
@set PROTOCOL_HANDERS=%PROTOCOL_HANDERS%;
    com.edocs.domain.telco.lde.protocol

```

Line continued

```

@set JAVA_OPTIONS=%JAVA_OPTIONS% -Dedx.home="%EDX_HOME%"
-Djava.protocol.handler.pkgs=%PROTOCOL_HANDLERS

```

Line continued

5. Define View

The LDE protocol is specified via the java.protocol.handler.pkgs property.

Example Tiles definition for a LDE based view:

```
<definition name=".my.ldebased.view" extends=".view.setup">
  <put name="viewName" value="viewName"/>
  <put name="viewType" value="HTML"/>
  <put name="ddn" value="associatedDDN"/>
  <put name="useExistingViewInfo" value="true"/>
</definition>
```

Must be true

{WEBAPPLICATION}/WEB-INF/tiles-*.xml

4

Security: Enrollment, Authentication, Authorization

This chapter describes:

- Authentication concepts
- Role-based security concepts
- Implementing access controls
- Implementing custom security providers

4.1 About Role-Based Security

Security is based on:

- Authentication - The process where by a user is validated, typically using a uid/pwd combination
- Authorization - The process of determining whether an authenticated user can access a resource
- Access Rights - The definition who has the rights to access a specific resource

TBM Provides:

- Security Authorization framework
- Security SPIs

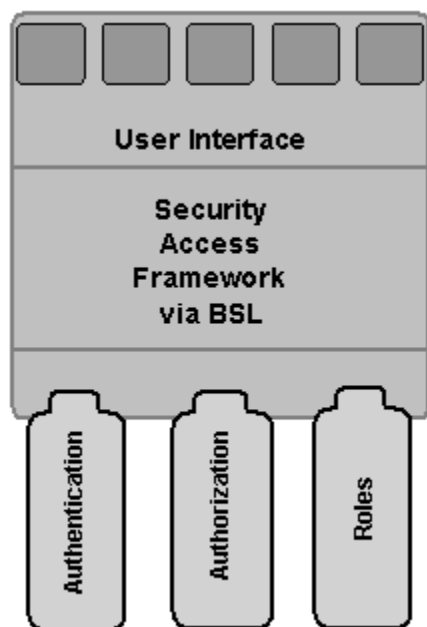
The edocs Security Authorization Framework provides:

- Definitions of access rights via an XML or custom store
- Run time checking access

The edocs BSL layer provides:

- IAuthenticationManager - Authenticate users

- ISecurityProfileManager - Create and otherwise manage user credentials & security questions
- IUserProfileManager - Manage userprofile data such name, address.



4.1.1 What is Security and Authorization Framework (SAF)?

The Security and Authorization Framework (SAF) provides the answer to the question:

- Given user, resource, and request context, should application grant or deny resource access

The SAF subsystem:

- Takes users, and resources and determines access rights based on a pluggable rules based authorization policy system
- Defaults to an XML based authorization scheme packaged with TBM
- Uses the underlying security management framework to obtain user and role information

See the SAF javadoc delivered with this SDK for details about using and customizing the Security and Authorization Framework.

4.1.2 Authenticate

TBM Authentication Subsystem:

- Includes authentication and impersonate support
- Return ISubject instances
- Defined by IAuthenticationManager

ISubject instances contain user* and role data for a given authenticated user

* Similar to Principals for those familiar with JAAS

4.1.3 Security Profile

TBM Security Profile Subsystem:

- Manages ISecurityProfile
- Manages ISecurityRole
- Defined by ISecurityProfileManager

ISecurityProfile instances contain roles', secret questions and similar information for a given uid.

ISecurityRole instances contain role and role descriptions

4.1.4 User Profiles

TBM User Profile Subsystem:

- Manages IUserProfile
- Defined by IUserProfileManager

IUserProfile instances contain name,address, email etc information for a given uid.

4.1.5 Security Management

Implementing Security Management involves:

- Implementing authorization
- Implementing Security Profiles including roles
- Implementing User Profiles*
- Linking common sub-systems

* User profiles are not strictly required but add value

4.1.6 Authorization

Implementing authorization involves:

- Creating and registering permissions on resources
- Authorizing at run time by:
 1. Obtaining ISubject instance from the BSL subsystem
 2. Obtaining permission instances for a given resource
 3. Testing access via the SAF Subsystem

4.2 Controlling Access to Resources

4.2.1 Using the SAF Sub-system

The SAF sub-system can be used with:

- Action classes to control access to resources
- In EdocsBaseMenuItem extended classes to control menu element visibility
- In any program element where a resource access should be protected

RITilesMenuBean can be used out of the box to provide security on Struts actions via SAF

4.2.2 SAF Components

SAF subsystem requires:

- Resource information - name etc of resource being accessed
- Roles - Defining sets of users
- ISubject - Instance of authorized subject attempting to access resource
- Permission object - Instance of a permission defined on the resource
- Policy files - Linking a given resource to a set of authorization rules

4.2.3 Policy Store

The SAF default policy store:

- Is defined in XML
- Contains permission elements defining a mapping between a right to access resources with a set of authorization rules

Permission elements contain:

- <name> - thing to protect</name>
- <cpath> - permission object </cpath>
- <rule> rules on who can access resources
 - <name> </name>
 - <type>SecurityRole</type>
 - <value>Comma separated list of security roles</value>

Permissions

- Are based on class implementations
- May imply one another to implement hierarchical rights

Policy Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE policystore PUBLIC .../>
<policystore>
  <permission>
    <name>SuperAdminSection</name>
    <cpath>com.edocs. . . ExtendsBasePermission</cpath>
    <rule>
      <name>all</name>
      <type>SecurityRole</type>
      <values>SUPER_ADMIN</values>
    </rule>
  </permission>
  ...
</policystore>
```

WEB-INF\classes\azcfg\policy\azpolicy.xml

Permission Example:

```
<permission>
  <name>special.txt</name>
  <cpath>com.MyPermission</cpath>

  <rule>
    <name>adm</name>
    <type>SecurityRole</type>
    <values>SPECIAL_ONLY
    </values>
  </rule>
</permission>
```

What are we protecting?

Who can access it?
Comma separated list

Roles come from the
underlying `ISecurityProfileManager`
implementation

4.2.4 Rights Engine

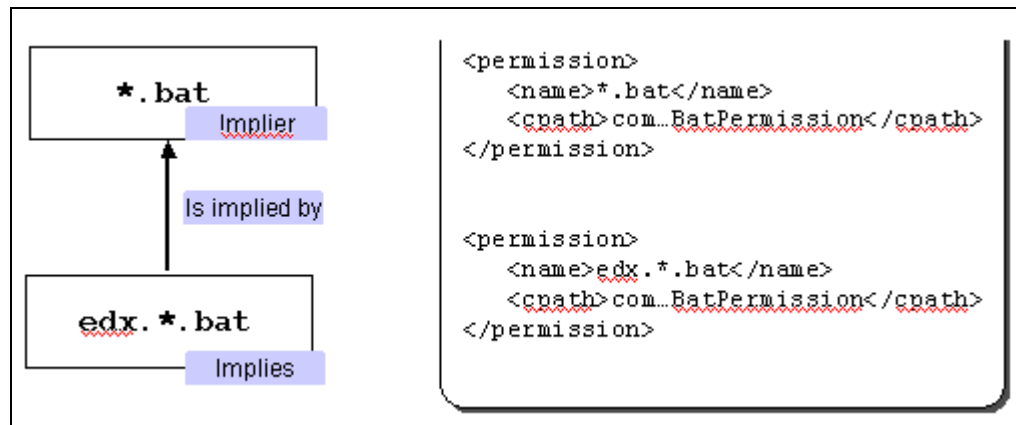
To check access rights via the access engine:

1. Obtains all rules defined for a permission request
2. Given a permission, get all mapped policy rules
 - If no mapped policy rules found
 1. Get implied permission
 2. Continue
3. For mapped policy rules check
 1. Does user have role granting access? Allow.
 2. Otherwise deny

4.2.5 Implied Policies

Policies can be explicit or imply one another.

Permission classes can parse resource name and return "implied" permissions.



4.2.6 Permission Classes

- Model resource types (eg. JSP, Text file, URL)
- Class instances instantiated with a particular name model resource instances
- Are extensions of `PermissionBase` which includes core methods such as `equals` and `hashCode`
- Can return "impliers"

4.2.7 Custom Impliers

Custom impliers are used when one permission may imply another.

edx.*.bat is implied by *.bat:

```

public final class BatPermission
{
    . . .
    public IPermission getImplier() throws AZException {
        IPermission perm = null;
        String name = this.getName();
        // If name does not match the pattern edx.*.bat
        // check against *.bat
        if (!name.startsWith("edx.*") && name.endsWith(".bat"))
            perm = new BatPermission("*.bat");
        }
        return perm;
    }
}

```

4.2.8 JSP Permission - Example

This permission can be used with policies of the form:

```

<permission>
  <name>*.jsp</name>
  <cpath>JSPPermission</cpath>
  <rule>
    <name>all</name>
    <type>SecurityRole</type>
    <values>Everyone</values>
  </rule>
</permission>
<permission>
  <name>admin.jsp</name>
  <cpath>JSPPermission</cpath>
  <rule>
    <name>all</name>
    <type>SecurityRole</type>
    <values>ADMIN</values>
  </rule>
</permission>

```

To grant access to most jsps but guard some.

```

import com.edocs.ps.security.authorization.AZException;
import com.edocs.ps.security.authorization.IPermission;

public final class JSPPermission
    extends PermissionBase
    implements IPermission {

    private static final String ANY_JSP = "*.jsp";
    private static final String JSP_EXT = ".jsp";

    public JSPPermission(String name) {
        super(name);
    }
    /**
     * Limited to matching * as 'all'.
     */
    public IPermission getImplier() throws AZException {
        IPermission perm = null;
        String name = this.getName();
        if (!name.equals(ANY_JSP) && name.endsWith(JSP_EXT)) {
            perm = new JSPPermission(ANY_JSP);
        }
        return perm;
    }
}

```

4.2.9 Using programmatic security

Steps for using programmatic security:

1. Write a permission that extends BasePermission. (See details, below.)
 - a. Implement getImplier() if required
2. Register the new permissions in the azpolicy.xml. (See details, below.)
3. Check policy where applicable:(e.g. in a action)
 - a. Obtain an ISubject
 - b. Create an instance of a permission representing the resource
 - c. Use SAF to check the access

1. Write permission

Permission to check access on edocs configuration files:

```
import com.edocs.ps.security.authorization.AZException;
import com.edocs.ps.security.authorization.IPermission;

public final class BatPermission extends PermissionBase implements IPermission {
    public BatPermission (String name) {
        super(name);
    }

    public IPermission getImplier() throws AZException {
        IPermission perm = null;
        String name = this.getName();
        if (!name.startsWith("edx.*") && name.endsWith(".bat")) {
            perm = new BatPermission ("*.bat");
        }
        return perm;
    }
}
```

2. Register permission

Permission Example:

```
<permission>
  <name>*.bat</name>
  <cpath>batPermission</cpath>
  <rule>
    <name>adm</name>
    <type>SecurityRole</type>
    <values>EveryoneRole</values>
  </rule>
</permission>
<permission>
  <name>edx.config.bat</name>
  <cpath>batPermission</cpath>
  <rule>
    <name>adm</name>
    <type>SecurityRole</type>
    <values>ADMIN</values>
  </rule>
</permission>
```

Obtain an ISubject

ISubject instances can be obtained from using SessionUtils.

Obtaining an ISubject within a Action Class:

```
import com.edocs.common.api.bsl.ISession;
import com.edocs.ps.security.authentication.ISubject;
import com.edocs.ps.security.authentication.SubjectUtils;

import javax.servlet.http.HttpServletRequest;

. . .

ISubject subject = ISession.getUserSubject (HttpServletRequest);
```

Check permissions method

Sample Check Permissions Method:

```
import com.edocs.ns.security.authorization.*;

public boolean checkPermission(ISubject subject,
                              IPermission permission) {

    boolean permitted = false;
    try {
        AZContext ctx = new AZContext();
        permitted = AZEngine.checkAccess(perm, subject, ctx);
    } catch (AZException e) {
        log.error("Error checking permission.", e);
    }
    return permitted;
}
```

3. Check access

Sample Check Permissions Method:

```
ISubject subject =
SessionUtils.getUserSubject(HttpServletRequest);

Permission perm = new BatPermission("edx.config.bat");

boolean allowed = checkPermission(subject, perm);
if (allowed)
    // you may change config settings
else
    // you may NOT change config settings
```

4.3 Integrating Security Providers

This section describes how to integrate security interfaces from TBM's Business Services Layer (BSL):

- Working with IAuthenticationManager
- Working with ISecurityProfile
- Working with ISecurityProfileManager

4.3.1 BSL Security SPI's

The TBM Business Services Layer provides: Service Provider Interfaces (SPI's) for the purposes of providing custom:

- Authorization
- Security and user profile management

- Company and account management

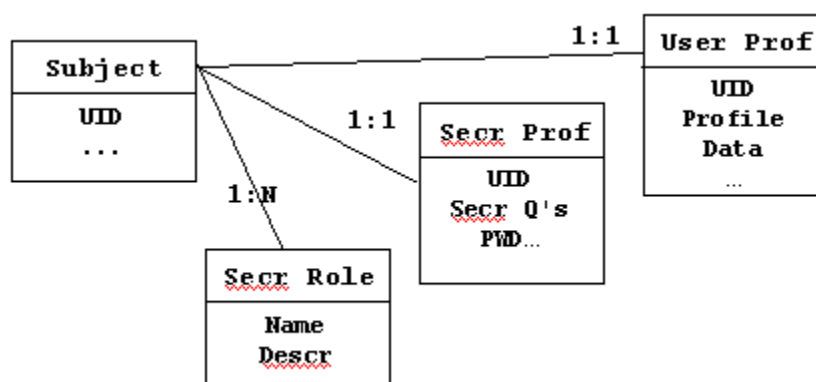
Authorization, Security and User profile management are considered Security management SPI's and are used to implement custom Single Sign-On etc.

See the BSL javadocs delivered with this SDK for implementation details about the BSL API and SPI.

4.3.2 BSL Security Concepts

BSL Security provides the instances of the classes required to answer the question: Given user, resource, and request context, should application grant or deny resource access

BSL is conceptually modeled as:



Subject:

- Modeled by ISubject
- Managed by IAuthenticationManager

Roles and Security Profiles:

- Modeled by ISecurityRole, ISecurityProfile
- Managed by IAuthenticationManager

User Profiles:

- Modeled by IUserProfile
- Managed by IUserProfileManager

4.3.3 Common Implementation Use Cases

Federate with existing authentication only:

- Only need implement IAuthenticationManager
- Cannot self-enroll, can manage security questions and user profile data
- User profile and security profile handled by default systems

Authentication and account management

- Implement `IAuthenticationManager`
- Implement `ISecurityProfileManager`
- Can self-enroll, new accounts populated back to system of record

Authentication, account management, and user profile

- Implement `IAuthenticationManager`, `ISecurityProfileManager` and `IUserProfileManager`
- Can self-enroll, new accounts populated back to system of record
- Can access pre-populated specialized User Profile data

4.3.4 Authentication

`IAuthenticationManager` interface implementations provide authorization services to BSL

`IAuthenticationManager` Methods:

```
public ISubject authenticate(String uid,  
                             String password);
```

Given a UID/pwd
Return an ISubject if
the combination is valid

```
public ISubject impersonate(String uid)
```

Assuming you are a valid
CSR, return a ISubject
reflecting the provided UID

`ISubject` implementations contain:

- Principal - "who" the subject is
- Principal roles - "what roles the subject belongs to"

4.3.5 Security Manager Profiles

`ISecurityProfileManager` interface implementations provide role and security profile services to BSL

ISecurityProfile ISecurityProfileManager Methods (partial list) :

<code>public ISecurityProfile createSecurityProfile(String uid, String password);</code>	Create a new "user"
<code>public ISecurityProfile updateSecurityProfile (ISecurityProfile profile)</code>	Update a "user"
<code>public void removeSecurityProfile(ISecurityProfile profile)</code>	Delete a "user"
<code>public ISecurityProfile[] getSecurityProfile(IFilter filter);</code>	Get some "users"

ISecurityRole ISecurityProfileManager Methods :

<code>public ISecurityRole createRole(String name, String description);</code>	Create a role
<code>public ISecurityRole getRole(String name) ;</code>	Find a role
<code>ISecurityRole[] getAllRoles();</code>	Get all roles

4.3.6 User Profiles

IUserProfileManager interface implementations provide user profile services to BSL:

<code>IUserProfile IUserProfileManager</code> Methods (partial list):	Create a new user profile
<code>public IUserProfile createUserProfile(String uid,...);</code>	
<code>public void removeUserProfile(IUserProfile);</code>	Remove a user profile
<code>public IUserProfile updateUserProfile(IUserProfile);</code>	Update a user profile
<code>public IUserProfile[] getUserProfile(IFilter filter)</code>	Get some user profiles

4.3.7 Configuring BSL Providers

The TBM BSL default implementations can be replaced via custom factory classes.

Factories are defined via system properties.

5

Customer Service Representative (CSR) Access/Capabilities

5.1 Overview

The CSR application delivered with TBM provides an interface to create and manage CSR administrators and organizations. Through this application a CSR can also impersonate a user. When a CSR enrolls in a CSR-enabled application, profiles are created in the database just as they are when regular users enroll. Depending on the CSR roles configured, a CSR can be limited to specific UI views and actions on behalf of another user. As with normal users, SAF authorizes access based on the permissions set for the CSR role.

5.2 CSR Access (Impersonate User)

When a CSR logs into the CSR application, he/she intends to either administer organizations, search for users or other CSR's or impersonate another user to provide support for that user. A CSR can see all the users he works with and click on an impersonate hyperlink for that user. When the impersonate hyperlink is clicked, `CSRAction.impersonate()` is executed:

```
public class CSRAction extends EdocsAction {
    private static Log log = LogFactory.getLog(CSRAction.class);

    public ActionForward impersonate(String uid,
                                    ActionMapping mapping,
                                    HttpServletRequest request,
                                    HttpServletResponse response) {
        if (log.isDebugEnabled()) {
            log.debug("Doing CSRAction: Impersonate");
        }
        ActionErrors errors = null;
```

```

        try {
            String urlString = "uid=" + uid
                                + "&pwd=" +
SessionUtils.getUserSession(request).getUser().getSecurityProfile().
getPassword()
                                + "&csr=" + SessionUtils.getUserId(request);

            if (log.isDebugEnabled()) {
                log.debug("Impersonate: urlString contains uid: " + uid + " csr:"
+ SessionUtils.getUserId(request));
            }

            // DDN needs to be passed separately in the URL String. Else, payment
            // initialization will fail (really!)
            // TODO : figure out why on UNIX this call to getConfig gets a value
            not in the properties file or the source file
            String link = "/tbmb/impersonateuser.do" +
//CSRConfiguration.getConfig().getCustAppURL() +
                "?value(" + RIConstants.CSR_TOKEN_ATTRIBUTE + ")="
                + RIEncryptUtil.encrypt(urlString)
                + "&ddn=" + CSRConfiguration.getConfig().getCustAppDdn();

            if (log.isDebugEnabled()) {
                log.debug("@impersonateCustomer: Set Link request attribute to :
" +
                link);
            }

            // Write a record to the EaDirect log to indicate that the
            // user was impersonated
            Logger.log(new CSRActivityItem("TBM_CSR",
                "Impersonate",
                CSRConfiguration.getConfig().getCustAppDdn(),
                SessionUtils.getUserId(request), // CSR User Id.
                "",
                "",
                "STARTING",
                uid, // Log userid of user being impersonated.
                "",
                "",
                ""));

            // Use request.setAttribute() so that the parameters do not show up
            // in the URL.

```

```

        request.setAttribute("CSRImpersonationUrl", link);
        return mapping.findForward(RIConstants.SUCCESS);
    } catch (EncryptionException e) {
        log.error("EncryptionException: " + e.getMessage(), e);
        errors.add(ActionErrors.GLOBAL_ERROR, new
            ActionError(RIHierarchyMessages.ENCRYPTION_EXCEPTION));
    } catch (RIException e) {
        log.error("Exception with RI: " + e.getMessage(), e);
        errors.add(ActionErrors.GLOBAL_ERROR, new
            ActionError(RIHierarchyMessages.SYSTEM_RI_ERROR));
    }
    if (!errors.isEmpty())
        saveErrors(request, errors);
    return mapping.findForward(RIConstants.ERROR); }

```

The CSRImpersonationUrl request attribute above is configured in app-config.properties. to the tbm-app with the authenticated user's credentials is passed. Read the tbm-csr-app javadocs for more detailed information about the CSR application.

BSL and SAF provide the authentication classes needed to impersonate. Implement BSL's IAuthenticationManager.impersonate(UID) to authenticate both the CSR and the impersonated user (UID). Configure the CSR permissions in the SAF policy file and create permission objects that the SAF engine uses to authenticate and authorize user access.

To create tiered access for CSR's, configure the SAF policy file with permissions tied to specific roles, such as CSR SuperAdmin (all privileges), CSR Manager (most privileges), CSR (some privileges), CSRImpersonate (impersonation only). Create permission classes to filter access. Employ the checkAccess tile definition in UI.jsp's to ask the SAF engine for access control by role (RBAC) of specific UI components.

For details about configuring SAF, see the Security chapter in this guide.

5.3 CSR Application

See the J2EE/tbm-csr-app javadoc for a description of the delivered reference implementation that demonstrates CSR access.

The CSR application WAR file contains the tiles (*.jsp) for the application. Under src/webroot are a variety of packages containing tiles that address key CSR view functions such as impersonating and finding a CSR's customer (*access-cust*), enrolling the CSR and searching for a customer's CSR (*manage-csr*), enrolling the customer (*manage-cust*), and searching and managing organizations (*manage-org*).

Under src/main/com/edocs/application/tbm/csr/ (compiled source) are action, form, and tag classes which comprise the model and controller of the CSR application. The common package contains a variety of CSR helper classes for login, enrollment, authentication, and configuration.

See the src/webroot/WEB-INF for the struts configuration beans and forwarding actions for this CSR application. The tiles configuration resides here also.

See the misc/app-config.properties file for how to configure access to the customer application from the CSR app, how to specify the customer application DDN that will be used during impersonation, and the list of CSR roles that are enabled.

```
#####  
# Below are CSR Application properties  
#####  
  
# URL to use to access the customer application  
tbm.csr.config.CUSTOMER_APPLICATION_URL=/tbm/impersonateuser.do  
  
# Customer Application DDN that will be used during impersonation  
tbm.csr.config.CUSTOMER_APPLICATION_DDN=TBM  
  
# List of CSR Roles  
tbm.csr.config.CSR_ROLE_LIST=Admin,CSR,SuperAdmin
```

Payment Cartridges

The default implementation delivered with TBM provides a payment component which provides a rich API for accessing e-Payment services. Read the javadocs delivered for this component and customize the key classes in this implementation to suit your payment needs.

Look in `tbm-app/src/webroot/payment` to find default jsp tile pages which you can re-use and customize for your own payment UI.

Look in `tbm-app/src/webroot/WEB-INF` to find the payment tiles and struts definition files for the default implementation of TBM. Examine these files to see how the UI is configured, what beans are available, and what payment actions or services are at your disposal.

Read the javadocs delivered for the TBM payment component in `com.edocs.common.api.payment`. Implement the payment interfaces as needed to customize the payment service. Place new implementations in `com.edocs.payment`. Re-use or extend already provided default implementation classes in `com.edocs.payment` as needed to customize the payment service for your application. Services, cassettes, and enrollment classes always reside in `com.edocs.payment` sub-packages.

New or template payment cartridges may be implemented to support international debit cards or other payment vehicles. To start, familiarize yourself with e-Payment cartridges by reading the chapter, *Implementing a Custom Payment Cartridge*, in the *Payment Developer's Guide*. Here you learn how to create and integrate custom cartridges and how to configure these new cartridges through Command Center Payment Settings. *Implementing a Custom Payment Cartridge* references demonstration source code. See `cassette_demo.zip`, which is delivered with this SDK



Downloading Views and Converting Advanced Function Print Files (AFP) to PDF

7.1 Downloading CSV, XML, and PDF Views

TBM supports downloading of views in a variety of formats: csv, xml, and pdf. The default implementation, `tbm-b2b-app`, provides several examples of how to do this. Read the javadocs for `tbm-b2b-app` and components (`core/tags`, `PDF`, and `PDFTransformer`) for implementation details. Also review the tiles definitions, tag library definitions, and struts configuration files for downloading configuration (*`tbm/lib/tbmtags.tld`*, *`src/webroot/WEB-INF/struts_config.xml`*, *`tiles-defs-download.xml`*, *`src/webroot/_includes/taglibraries.jsp`*).

Downloadable views and reports UI pages include the tile, *`src/webroot/_includes/action.jsp`*, which renders a drop-down for downloads according to view type using the form bean, `com.edocs.application.tbm.b2b.forms.DownloadForm`. The PDF download is only available for account summaries and wireless number summaries. CSV and XML is available in all cases. The action on download selection is performed by `com.edocs.application.tbm.b2b.actions.DownloadAction`, which forwards to a handler for csv, xml, or pdf output.

The download section of `struts.config` contains the following view and report download action mappings:

```
<action path="/view/download"
type="com.edocs.application.tbm.b2b.actions.DownloadAction" name="downloadForm"
scope="request" validate="false">

    <forward name="/view/account_summary.do"
path="/view/download/acctSummaryDownload.do"/>

    <forward name="/view/wireless_number_summary.do"
path="/view/download/mtnSummaryDownload.do"/>

    <forward name="/view/detailed_usage_home_exp.do"
path="/view/download/callDetailHomeDownload.do"/>

    <forward name="/view/detailed_usage.do"
path="/view/download/callDetailHomeDownload.do"/>
```

```

        <forward name="/view/detailed_usage_roam_exp.do"
path="/view/download/callDetailRoamDownload.do"/>

        <forward name="/view/detailed_usage_roam_col.do"
path="/view/download/callDetailRoamDownload.do"/>

        <forward name="/view/reports.do"
path="/view/download/accountRptDownload.do"/>

        <forward name="/view/accountReport.do"
path="/view/download/accountRptDownload.do"/>

        <forward name="/view/wirelessNumReport.do"
path="/view/download/mtnRptDownload.do"/>

        <forward name="/view/printerFriendlyAccRpt"
path="/view/download/accountRptDownload.do"/>

        <forward name="/view/printerFriendlyMTNRpt"
path="/view/download/mtnRptDownload.do"/>

    </action>

    <action path="/view/download/acctSummaryDowload"
type="com.edocs.application.tbm.b2b.actions.DownloadAction">

        <forward name="pdf" path=".download.accountSummary.pdf"/>

        <forward name="csv" path=".download.accountSummary.csv"/>

        <forward name="xml" path=".download.accountSummary.xml"/>

    </action>

    <action path="/view/download/mtnSummaryDownload"
type="com.edocs.application.tbm.b2b.actions.DownloadAction">

        <forward name="pdf" path=".download.mtnSummary.pdf"/>

        <forward name="csv" path=".download.mtnSummary.csv"/>

        <forward name="xml" path=".download.mtnSummary.xml"/>

    </action>

    <action path="/view/download/callDetailHomeDownload"
type="com.edocs.application.tbm.b2b.actions.DownloadAction">

        <forward name="pdf" path=""/>

        <forward name="csv" path=".download.callDetailHome.csv"/>

        <forward name="xml" path=".download.callDetailHome.xml"/>

    </action>

    <action path="/view/download/callDetailRoamDownload"
type="com.edocs.application.tbm.b2b.actions.DownloadAction">

        <forward name="pdf" path=""/>

        <forward name="csv" path=".download.callDetailRoam.csv"/>

        <forward name="xml" path=".download.callDetailRoam.xml"/>

    </action>

    <action path="/view/download/accountRptDownload"
type="com.edocs.application.tbm.b2b.actions.ReportDownloadAction"
parameter="ACCOUNT_REPORT">

        <forward name="default"
path="/view/download/b2bLongCalls.do"/>

        <forward name="B2B_LongCall"
path="/view/download/b2bLongCalls.do"/>

        <forward name="B2B_ExpCall"
path="/view/download/b2bExpCalls.do"/>

        <forward name="B2B_CallNum"
path="/view/download/b2bCallNum.do"/>

```

```

                <forward name="B2B_ExpCallNum"
path="/view/downloadExpCallNum.do"/>

                <forward name="B2B_CallCountry"
path="/view/download/b2bCallCountry.do"/>

                <forward name="B2B_SumType"
path="/view/download/b2bSumType.do"/>

                <forward name="B2B_SumMonth"
path="/view/download/b2bSumMonth.do"/>

                <forward name="B2B_SumTime"
path="/view/download/b2bSumTime.do"/>

            </action>

            <action path="/view/download/b2bLongCalls"
type="com.edocs.application.tbm.b2b.actions.ReportDownloadAction"
parameter="FIND_DOWNLOAD_VIEW">

                <forward name="csv"
path="download.reports.accountReports.xLongestCalls.csv.view"/>

                <forward name="xml"
path="download.reports.accountReports.xLongestCalls.xml.view"/>

            </action>

            <action path="/view/download/b2bExpCalls"
type="com.edocs.application.tbm.b2b.actions.ReportDownloadAction"
parameter="FIND_DOWNLOAD_VIEW">

                <forward name="csv"
path="download.reports.accountReports.xMostExpCalls.csv.view"/>

                <forward name="xml"
path="download.reports.accountReports.xMostExpCalls.xml.view"/>

            </action>

            <action path="/view/download/b2bCallNum"
type="com.edocs.application.tbm.b2b.actions.ReportDownloadAction"
parameter="FIND_DOWNLOAD_VIEW">

                <forward name="csv"
path="download.reports.accountReports.xMostCalledNumbers.csv.view"/>

                <forward name="xml"
path="download.reports.accountReports.xMostCalledNumbers.xml.view"/>

            </action>

            <action path="/view/downloadExpCallNum"
type="com.edocs.application.tbm.b2b.actions.ReportDownloadAction"
parameter="FIND_DOWNLOAD_VIEW">

                <forward name="csv"
path="download.reports.accountReports.xMostExpCalledNos.csv.view"/>

                <forward name="xml"
path="download.reports.accountReports.xMostExpCalledNos.xml.view"/>

            </action>

            <action path="/view/download/b2bCallCountry"
type="com.edocs.application.tbm.b2b.actions.ReportDownloadAction"
parameter="FIND_DOWNLOAD_VIEW">

                <forward name="csv"
path="download.reports.accountReports.xMostCalledCountries.csv.view"/>

                <forward name="xml"
path="download.reports.accountReports.xMostCalledCountries.xml.view"/>

            </action>

            <action path="/view/download/b2bSumType"
type="com.edocs.application.tbm.b2b.actions.ReportDownloadAction"
parameter="FIND_DOWNLOAD_VIEW">

```

```

        <forward name="csv"
path="download.reports.accountReports.costSummaryByCallType.csv.view"/>
        <forward name="xml"
path="download.reports.accountReports.costSummaryByCallType.xml.view"/>
    </action>

    <action path="/view/download/b2bSumMonth"
type="com.edocs.application.tbm.b2b.actions.ReportDownloadAction"
parameter="FIND_DOWNLOAD_VIEW">
        <forward name="csv"
path="download.reports.accountReports.costSummaryByMonth.csv.view"/>
        <forward name="xml"
path="download.reports.accountReports.costSummaryByMonth.xml.view"/>
    </action>

    <action path="/view/download/b2bSumTime"
type="com.edocs.application.tbm.b2b.actions.ReportDownloadAction"
parameter="FIND_DOWNLOAD_VIEW">
        <forward name="csv"
path="download.reports.accountReports.costSummaryByTimePeriod.csv.view"/>
        <forward name="xml"
path="download.reports.accountReports.costSummaryByTimePeriod.xml.view"/>
    </action>

    <action path="/view/download/mtnRptDownload"
type="com.edocs.application.tbm.b2b.actions.ReportDownloadAction"
parameter="MTN_REPORT">
        <forward name="default"
path="/view/download/mtnB2bTopTen.do"/>
        <forward name="B2B_TopTen"
path="/view/download/mtnB2bTopTen.do"/>
        <forward name="B2B_CallCalls"
path="/view/download/mtnB2bCallCalls.do"/>
        <forward name="B2B_CallChgs"
path="/view/download/mtnB2bCallChgs.do"/>
        <forward name="B2B_CallMins"
path="/view/download/mtnB2bCallMins.do"/>
        <forward name="B2B_DestCalls"
path="/view/download/mtnB2bDestCalls.do"/>
        <forward name="B2B_DestChgs"
path="/view/download/mtnB2bDestChgs.do"/>
        <forward name="B2B_DestMins"
path="/view/download/mtnB2bDestMins.do"/>
        <forward name="B2B_OriginCalls"
path="/view/download/mtnB2bOriginCalls.do"/>
        <forward name="B2B_OriginChgs"
path="/view/download/mtnB2bOriginChrgs.do"/>
        <forward name="B2B_OriginMins"
path="/view/download/mtnB2bOriginMins.do"/>
        <forward name="B2B_RateCalls"
path="/view/download/mtnB2bRateCalls.do"/>
        <forward name="B2B_RateChgs"
path="/view/download/mtnB2bRateChgs.do"/>
        <forward name="B2B_RateMins"
path="/view/download/mtnB2bRateMins.do"/>
    </action>

```

```

        <action path="/view/download/mtnB2bTopTen"
type="com.edocs.application.tbm.b2b.actions.ReportDownloadAction"
parameter="FIND_DOWNLOAD_VIEW">
            <forward name="csv"
path="download.reports.wirelessNumberReport.top10MostExpCalls.csv.view"/>
            <forward name="xml"
path="download.reports.wirelessNumberReport.top10MostExpCalls.xml.view"/>
        </action>

        <action path="/view/download/mtnB2bCallCalls"
type="com.edocs.application.tbm.b2b.actions.ReportDownloadAction"
parameter="FIND_DOWNLOAD_VIEW">
            <forward name="csv"
path="download.reports.wirelessNumberReport.callsByCallType.csv.view"/>
            <forward name="xml"
path="download.reports.wirelessNumberReport.callsByCallType.xml.view"/>
        </action>

        <action path="/view/download/mtnB2bCallChgs"
type="com.edocs.application.tbm.b2b.actions.ReportDownloadAction"
parameter="FIND_DOWNLOAD_VIEW">
            <forward name="csv"
path="download.reports.wirelessNumberReport.chargesByCallType.csv.view"/>
            <forward name="xml"
path="download.reports.wirelessNumberReport.chargesByCallType.xml.view"/>
        </action>

        <action path="/view/download/mtnB2bCallMins"
type="com.edocs.application.tbm.b2b.actions.ReportDownloadAction"
parameter="FIND_DOWNLOAD_VIEW">
            <forward name="csv"
path="download.reports.wirelessNumberReport.minutesByCallType.csv.view"/>
            <forward name="xml"
path="download.reports.wirelessNumberReport.minutesByCallType.xml.view"/>
        </action>

        <action path="/view/download/mtnB2bDestCalls"
type="com.edocs.application.tbm.b2b.actions.ReportDownloadAction"
parameter="FIND_DOWNLOAD_VIEW">
            <forward name="csv"
path="download.reports.wirelessNumberReport.callsByDestination.csv.view"/>
            <forward name="xml"
path="download.reports.wirelessNumberReport.callsByDestination.xml.view"/>
        </action>

        <action path="/view/download/mtnB2bDestChgs"
type="com.edocs.application.tbm.b2b.actions.ReportDownloadAction"
parameter="FIND_DOWNLOAD_VIEW">
            <forward name="csv"
path="download.reports.wirelessNumberReport.chargesByDestination.csv.view"/>
            <forward name="xml"
path="download.reports.wirelessNumberReport.chargesByDestination.xml.view"/>
        </action>

        <action path="/view/download/mtnB2bDestMins"
type="com.edocs.application.tbm.b2b.actions.ReportDownloadAction"
parameter="FIND_DOWNLOAD_VIEW">
            <forward name="csv"
path="download.reports.wirelessNumberReport.minutesByDestination.csv.view"/>

```

```

        <forward name="xml"
path="download.reports.wirelessNumberReport.minutesByDestination.xml.view"/>
    </action>

    <action path="/view/download/mtnB2bOriginCalls"
type="com.edocs.application.tbm.b2b.actions.ReportDownloadAction"
parameter="FIND_DOWNLOAD_VIEW">

        <forward name="csv"
path="download.reports.wirelessNumberReport.callsByCallOrigination.csv.view"/>

        <forward name="xml"
path="download.reports.wirelessNumberReport.callsByCallOrigination.xml.view"/>

    </action>

    <action path="/view/download/mtnB2bOriginChrgs"
type="com.edocs.application.tbm.b2b.actions.ReportDownloadAction"
parameter="FIND_DOWNLOAD_VIEW">

        <forward name="csv"
path=".reports.wirelessNumberReport.chargesByCallOrigination.csv.view"/>

        <forward name="xml"
path=".reports.wirelessNumberReport.chargesByCallOrigination.xml.view"/>

    </action>

    <action path="/view/download/mtnB2bOriginMins"
type="com.edocs.application.tbm.b2b.actions.ReportDownloadAction"
parameter="FIND_DOWNLOAD_VIEW">

        <forward name="csv"
path="download.reports.wirelessNumberReport.minutesByCallOrigination.csv.view"/>

        <forward name="xml"
path="download.reports.wirelessNumberReport.minutesByCallOrigination.xml.view"/>

    </action>

    <action path="/view/download/mtnB2bRateCalls"
type="com.edocs.application.tbm.b2b.actions.ReportDownloadAction"
parameter="FIND_DOWNLOAD_VIEW">

        <forward name="csv"
path="download.reports.wirelessNumberReport.callsByRatePeriod.csv.view"/>

        <forward name="xml"
path="download.reports.wirelessNumberReport.callsByRatePeriod.xml.view"/>

    </action>

    <action path="/view/download/mtnB2bRateChgs"
type="com.edocs.application.tbm.b2b.actions.ReportDownloadAction"
parameter="FIND_DOWNLOAD_VIEW">

        <forward name="csv"
path="download.reports.wirelessNumberReport.chargesByRatePeriod.csv.view"/>

        <forward name="xml"
path="download.reports.wirelessNumberReport.chargesByRatePeriod.xml.view"/>

    </action>

    <action path="/view/download/mtnB2bRateMins"
type="com.edocs.application.tbm.b2b.actions.ReportDownloadAction"
parameter="FIND_DOWNLOAD_VIEW">

        <forward name="csv"
path="download.reports.wirelessNumberReport.minutesByRatePeriod.csv.view"/>

        <forward name="xml"
path="download.reports.wirelessNumberReport.minutesByRatePeriod.xml.view"/>

    </action>

```

The tiles definitions for downloading look like this:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE tiles-definitions PUBLIC "-//Apache Software Foundation//DTD Tiles
Configuration 1.1/EN"
    "http://jakarta.apache.org/struts/dtds/tiles-config_1_1.dtd">
<tiles-definitions>
    <!-- ===== -->
    <!-- Definitions for downloads -->
    <!-- ===== -->
    <!--Download Views Setup-->
    <definition name=".download.setup"
path="/_templates/display_download_view.jsp"
controllerClass="com.edocs.application.tbm.displayview.DisplayTiledView">
        <put name="viewName" value="DefaultView"/>
        <put name="viewType" value="XSLT"/>
        <put name="ddn" value="TBM"/>
        <put name="download" value="true"/>
    </definition>
    <!-- Account Summary -->
    <definition name=".download.accountSummary.pdf"
extends=".download.setup">
        <put name="viewName" value="B2B_AcctSum_PDF"/>
        <put name="viewType" value="PDF_FO"/>
        <put name="ddn" value="TBM"/>
    </definition>
    <definition name=".download.accountSummary.csv"
extends=".download.setup">
        <put name="viewName" value="B2B_AcctSum_CSV"/>
        <put name="viewType" value="XSLT"/>
        <put name="ddn" value="TBM"/>
    </definition>
    <definition name=".download.accountSummary.xml"
extends=".download.setup">
        <put name="viewName" value="B2B_AcctSum_XML"/>
        <put name="viewType" value="XSLT"/>
        <put name="ddn" value="TBM"/>
    </definition>
    <!-- MTN Summary -->
    <definition name=".download.mtnSummary.pdf" extends=".download.setup">
        <put name="viewName" value="B2B_MTNSum_PDF"/>
        <put name="viewType" value="PDF_FO"/>
        <put name="ddn" value="TBM"/>
    </definition>
```

```

</definition>
<definition name=".download.mtnSummary.csv" extends=".download.setup">
    <put name="viewName" value="B2B_MTNSum_CSV"/>
    <put name="viewType" value="XSLT"/>
    <put name="ddn" value="TBM"/>
</definition>
<definition name=".download.mtnSummary.xml" extends=".download.setup">
    <put name="viewName" value="B2B_MTNSum_XML"/>
    <put name="viewType" value="XSLT"/>
    <put name="ddn" value="TBM"/>
</definition>
<!-- Call Detail Home -->
<definition name=".download.callDetailHome.csv"
extends=".download.setup">
    <put name="viewName" value="B2B_DetHome_CSV"/>
    <put name="viewType" value="XSLT"/>
    <put name="ddn" value="TBM"/>
</definition>
<definition name=".download.callDetailHome.xml"
extends=".download.setup">
    <put name="viewName" value="B2B_DetHome_XML"/>
    <put name="viewType" value="XSLT"/>
    <put name="ddn" value="TBM"/>
</definition>
<!-- Call Detail Roaming -->
<definition name=".download.callDetailRoam.csv"
extends=".download.setup">
    <put name="viewName" value="B2B_DetRoam_CSV"/>
    <put name="viewType" value="XSLT"/>
    <put name="ddn" value="TBM"/>
</definition>
<definition name=".download.callDetailRoam.xml"
extends=".download.setup">
    <put name="viewName" value="B2B_DetRoam_XML"/>
    <put name="viewType" value="XSLT"/>
    <put name="ddn" value="TBM"/>
</definition>
<!--
===== -->
<!-- ===== Account Report download definitions
===== -->
<!--
===== -->
<!-- X longest calls -->

```

```

        <definition
name="download.reports.accountReports.xLongestCalls.csv.view"
extends=".download.setup">
            <put name="viewName" value="B2B_LongCall_CSV"/>
            <put name="ddn" value="TBM_rpt"/>
        </definition>

        <definition
name="download.reports.accountReports.xLongestCalls.xml.view"
extends=".download.setup">
            <put name="viewName" value="B2B_LongCall_XML"/>
            <put name="ddn" value="TBM_rpt"/>
        </definition>

        <!--      X most expensive calls      -->

        <definition
name="download.reports.accountReports.xMostExpCalls.csv.view"
extends=".download.setup">
            <put name="viewName" value="B2B_ExpCall_CSV"/>
            <put name="ddn" value="TBM_rpt"/>
        </definition>

        <definition
name="download.reports.accountReports.xMostExpCalls.xml.view"
extends=".download.setup">
            <put name="viewName" value="B2B_ExpCall_XML"/>
            <put name="ddn" value="TBM_rpt"/>
        </definition>

        <!--      X most called numbers      -->

        <definition
name="download.reports.accountReports.xMostCalledNumbers.csv.view"
extends=".download.setup">
            <put name="viewName" value="B2B_CallNum_CSV"/>
            <put name="ddn" value="TBM_rpt"/>
        </definition>

        <definition
name="download.reports.accountReports.xMostCalledNumbers.xml.view"
extends=".download.setup">
            <put name="viewName" value="B2B_CallNum_XML"/>
            <put name="ddn" value="TBM_rpt"/>
        </definition>

        <!--      X most expensive called numbers      -->

        <definition
name="download.reports.accountReports.xMostExpCalledNos.csv.view"
extends=".download.setup">
            <put name="viewName" value="B2B_ExpCallNum_CSV"/>
            <put name="ddn" value="TBM_rpt"/>
        </definition>

```

```

        <definition
name="download.reports.accountReports.xMostExpCalledNos.xml.view"
extends=".download.setup">

        <put name="viewName" value="B2B_ExpCallNum_XML"/>
        <put name="ddn" value="TBM_rpt"/>

    </definition>

    <!--      X most called countries      -->

    <definition
name="download.reports.accountReports.xMostCalledCountries.csv.view"
extends=".download.setup">

        <put name="viewName" value="B2B_CallCountry_CSV"/>
        <put name="ddn" value="TBM_rpt"/>

    </definition>

    <definition
name="download.reports.accountReports.xMostCalledCountries.xml.view"
extends=".download.setup">

        <put name="viewName" value="B2B_CallCountry_XML"/>
        <put name="ddn" value="TBM_rpt"/>

    </definition>

    <!--      Cost summary by call type      -->

    <definition
name="download.reports.accountReports.costSummaryByCallType.csv.view"
extends=".download.setup">

        <put name="viewName" value="B2B_SumType_CSV"/>
        <put name="ddn" value="TBM_rpt"/>

    </definition>

    <definition
name="download.reports.accountReports.costSummaryByCallType.xml.view"
extends=".download.setup">

        <put name="viewName" value="B2B_SumType_XML"/>
        <put name="ddn" value="TBM_rpt"/>

    </definition>

    <!--      Cost summary by month      -->

    <definition
name="download.reports.accountReports.costSummaryByMonth.csv.view"
extends=".download.setup">

        <put name="viewName" value="B2B_SumMonth_CSV"/>
        <put name="ddn" value="TBM_rpt"/>

    </definition>

    <definition
name="download.reports.accountReports.costSummaryByMonth.xml.view"
extends=".download.setup">

        <put name="viewName" value="B2B_SumMonth_XML"/>
        <put name="ddn" value="TBM_rpt"/>

    </definition>

    <!--      Cost summary by time period      -->

```

```

        <definition
name="download.reports.accountReports.costSummaryByTimePeriod.csv.view"
extends=".download.setup">
            <put name="viewName" value="B2B_SumTime_CSV"/>
            <put name="ddn" value="TBM_rpt"/>
        </definition>

        <definition
name="download.reports.accountReports.costSummaryByTimePeriod.xml.view"
extends=".download.setup">
            <put name="viewName" value="B2B_SumTime_XML"/>
            <put name="ddn" value="TBM_rpt"/>
        </definition>

        <!--
===== -->

        <!-- ===== Wireless (MTNs) Report download defs =====
-->

        <!--
===== -->

        <!-- Top 10 Most Expensive Calls -->

        <definition
name="download.reports.wirelessNumberReport.top10MostExpCalls.csv.view"
extends=".download.setup">
            <put name="viewType" value="XSLT"/>
            <put name="viewName" value="B2B_TopTen_CSV"/>
            <put name="subIdx" value="true"/>
        </definition>

        <definition
name="download.reports.wirelessNumberReport.top10MostExpCalls.xml.view"
extends=".download.setup">
            <put name="viewType" value="XSLT"/>
            <put name="viewName" value="B2B_TopTen_XML"/>
            <put name="subIdx" value="true"/>
        </definition>

        <!-- Calls by Call Type -->

        <definition
name="download.reports.wirelessNumberReport.callsByCallType.csv.view"
extends=".download.setup">
            <put name="viewType" value="XSLT"/>
            <put name="viewName" value="B2B_CallCalls_CSV"/>
            <put name="subIdx" value="true"/>
        </definition>

        <definition
name="download.reports.wirelessNumberReport.callsByCallType.xml.view"
extends=".download.setup">
            <put name="viewType" value="XSLT"/>
            <put name="viewName" value="B2B_CallCalls_XML"/>
            <put name="subIdx" value="true"/>

```

```

</definition>

<!--      Charges by Call Type      -->

<definition
name="download.reports.wirelessNumberReport.chargesByCallType.csv.view"
extends=".download.setup">

    <put name="viewType" value="XSLT"/>
    <put name="viewName" value="B2B_CallChgs_CSV"/>
    <put name="subIdx" value="true"/>

</definition>

<definition
name="download.reports.wirelessNumberReport.chargesByCallType.xml.view"
extends=".download.setup">

    <put name="viewType" value="XSLT"/>
    <put name="viewName" value="B2B_CallChgs_XML"/>
    <put name="subIdx" value="true"/>

</definition>

<!--      Minutes by Call Type      -->

<definition
name="download.reports.wirelessNumberReport.minutesByCallType.csv.view"
extends=".download.setup">

    <put name="viewType" value="XSLT"/>
    <put name="viewName" value="B2B_CallMins_CSV"/>
    <put name="subIdx" value="true"/>

</definition>

<definition
name="download.reports.wirelessNumberReport.minutesByCallType.xml.view"
extends=".download.setup">

    <put name="viewType" value="XSLT"/>
    <put name="viewName" value="B2B_CallMins_XML"/>
    <put name="subIdx" value="true"/>

</definition>

<!--      Calls by Destination      -->

<definition
name="download.reports.wirelessNumberReport.callsByDestination.csv.view"
extends=".download.setup">

    <put name="viewType" value="XSLT"/>
    <put name="viewName" value="B2B_DestCalls_CSV"/>
    <put name="subIdx" value="true"/>

</definition>

<definition
name="download.reports.wirelessNumberReport.callsByDestination.xml.view"
extends=".download.setup">

    <put name="viewType" value="XSLT"/>
    <put name="viewName" value="B2B_DestCalls_XML"/>
    <put name="subIdx" value="true"/>

</definition>

```

```

<!-- Charges by Destination -->
<definition
name="download.reports.wirelessNumberReport.chargesByDestination.csv.view"
extends=".download.setup">
    <put name="viewType" value="XSLT"/>
    <put name="viewName" value="B2B_DestChgs_CSV"/>
    <put name="subIdx" value="true"/>
</definition>
<definition
name="download.reports.wirelessNumberReport.chargesByDestination.xml.view"
extends=".download.setup">
    <put name="viewType" value="XSLT"/>
    <put name="viewName" value="B2B_DestChgs_XML"/>
    <put name="subIdx" value="true"/>
</definition>
<!-- Minutes by Destination -->
<definition
name="download.reports.wirelessNumberReport.minutesByDestination.csv.view"
extends=".download.setup">
    <put name="viewType" value="XSLT"/>
    <put name="viewName" value="B2B_DestMins_CSV"/>
    <put name="subIdx" value="true"/>
</definition>
<definition
name="download.reports.wirelessNumberReport.minutesByDestination.xml.view"
extends=".download.setup">
    <put name="viewType" value="XSLT"/>
    <put name="viewName" value="B2B_DestMins_XML"/>
    <put name="subIdx" value="true"/>
</definition>
<!-- Calls by Call Origination -->
<definition
name="download.reports.wirelessNumberReport.callsByCallOrigination.csv.view"
extends=".download.setup">
    <put name="viewType" value="XSLT"/>
    <put name="viewName" value="B2B_OriginCalls_CSV"/>
    <put name="subIdx" value="true"/>
</definition>
<definition
name="download.reports.wirelessNumberReport.callsByCallOrigination.xml.view"
extends=".download.setup">
    <put name="viewType" value="XSLT"/>
    <put name="viewName" value="B2B_OriginCalls_XML"/>
    <put name="subIdx" value="true"/>
</definition>
<!-- Charges by Call Origination -->

```

```

        <definition
name=".reports.wirelessNumberReport.chargesByCallOrigination.csv.view"
extends=".download.setup">
            <put name="viewType" value="XSLT"/>
            <put name="viewName" value="B2B_OriginChgs_CSV"/>
            <put name="subIdx" value="true"/>
        </definition>

        <definition
name=".reports.wirelessNumberReport.chargesByCallOrigination.xml.view"
extends=".download.setup">
            <put name="viewType" value="XSLT"/>
            <put name="viewName" value="B2B_OriginChgs_XML"/>
            <put name="subIdx" value="true"/>
        </definition>

        <!-- Minutes by Call Origination -->

        <definition
name="download.reports.wirelessNumberReport.minutesByCallOrigination.csv.view"
" extends=".download.setup">
            <put name="viewType" value="XSLT"/>
            <put name="viewName" value="B2B_OriginMins_CSV"/>
            <put name="subIdx" value="true"/>
        </definition>

        <definition
name="download.reports.wirelessNumberReport.minutesByCallOrigination.xml.view"
" extends=".download.setup">
            <put name="viewType" value="XSLT"/>
            <put name="viewName" value="B2B_OriginMins_XML"/>
            <put name="subIdx" value="true"/>
        </definition>

        <!-- Calls by Rate Period -->

        <definition
name="download.reports.wirelessNumberReport.callsByRatePeriod.csv.view"
extends=".download.setup">
            <put name="viewType" value="XSLT"/>
            <put name="viewName" value="B2B_RateCalls_CSV"/>
            <put name="subIdx" value="true"/>
        </definition>

        <definition
name="download.reports.wirelessNumberReport.callsByRatePeriod.xml.view"
extends=".download.setup">
            <put name="viewType" value="XSLT"/>
            <put name="viewName" value="B2B_RateCalls_XML"/>
            <put name="subIdx" value="true"/>
        </definition>

        <!-- Charges by Rate Period -->

```

```

        <definition
name="download.reports.wirelessNumberReport.chargesByRatePeriod.csv.view"
extends=".download.setup">

        <put name="viewType" value="XSLT"/>

        <put name="viewName" value="B2B_RateChgs_CSV"/>

        <put name="subIdx" value="true"/>

    </definition>

    <definition
name="download.reports.wirelessNumberReport.chargesByRatePeriod.xml.view"
extends=".download.setup">

        <put name="viewType" value="XSLT"/>

        <put name="viewName" value="B2B_RateChgs_XML"/>

        <put name="subIdx" value="true"/>

    </definition>

    <!--      Minutes by Rate Period      -->

    <definition
name="download.reports.wirelessNumberReport.minutesByRatePeriod.csv.view"
extends=".download.setup">

        <put name="viewType" value="XSLT"/>

        <put name="viewName" value="B2B_RateMins_CSV"/>

        <put name="subIdx" value="true"/>

    </definition>

    <definition
name="download.reports.wirelessNumberReport.minutesByRatePeriod.xml.view"
extends=".download.setup">

        <put name="viewType" value="XSLT"/>

        <put name="viewName" value="B2B_RateMins_XML"/>

        <put name="subIdx" value="true"/>

    </definition>

    <!--      Reports definitions ends      -->

</tiles-definitions>
    
```

When a download is selected for one of the downloadable views, DownloadAction forwards to *src/webroot/_templates/display_download_view.jsp* (see the download.setup tile definition in *tiles-defs-download*). Display_download_view.jsp employs a tbmtag called, “getView”, which represents *com.edocs.application.tbm.core.tags.ViewDisplayTag*. ViewDisplayTag reads in view content type information using *com.edocs.application.tbm.core.common.RIViewProcessor*.

In all download cases, the controller for view presentation is *com.edocs.application.tbm.displayView.DisplayTiledView* (see the javadocs for the DisplayView component). DisplayTiledView uses the CAM layer to obtain the DDN view from e-Statement in the format prescribed by the tile definition above. (See *Section 3.1 View Architecture Basics*).

For PDF downloads, `ViewDisplayTag` employs the `edocs` component service, `PDFTransformer`, to enlist a third-party library (`CordaServer`) to transform and output the html view in pdf format. See the `PDFTransformer` component javadocs for implementation details.

For CSV and XML downloads, `ViewDisplayTag` simply writes the csv or xml view (as presented by e-Statement) to the output stream. (See the tile definitions for download views above.)

7.2 AFP-PDF Conversion

A Data Definition File, or DDF, contains the rules for finding and extracting data from an application's input data source. Normally you create one or more DDFs using e-Statement's `DefTool` exclusively for use with your organization's particular data source format and application. Advanced Function Printing Files (AFP) normally get converted to DDF's via e-Statement's `DefTool` which are then merged with HTML templates via e-Statement's `Composer` to generate an HTML dynamic view. In addition to HTML-formatted views, e-Statement application views can dynamically present data in other formats: CSV, XML, XSLT. These formats require generation of a DDF prior to conversion.

TBM now supports the PDF dynamic view type. To convert AFPs to PDF format, deploy the PDF component's `com.edocs.services.merger/pdf/PDFMerger.java`. `PDFMerger` extends `com.edocs.services.merger.BaseMerger` and implements `javax.ejb.SessionBean`.

The `PDFMerger` session bean directly converts an AFP input file to a PDF file and bypasses DDF generation. It employs a third-party library, `PDF/pdfMerger/lib/StreamA2P.jar`. See the `PDF/config/view-types.xml` for the configuration needed to support the PDF view-type. The e-Statement publisher will support a PDF output file view set generation, if the PDF view-type is configured properly in `view-types.xml`.

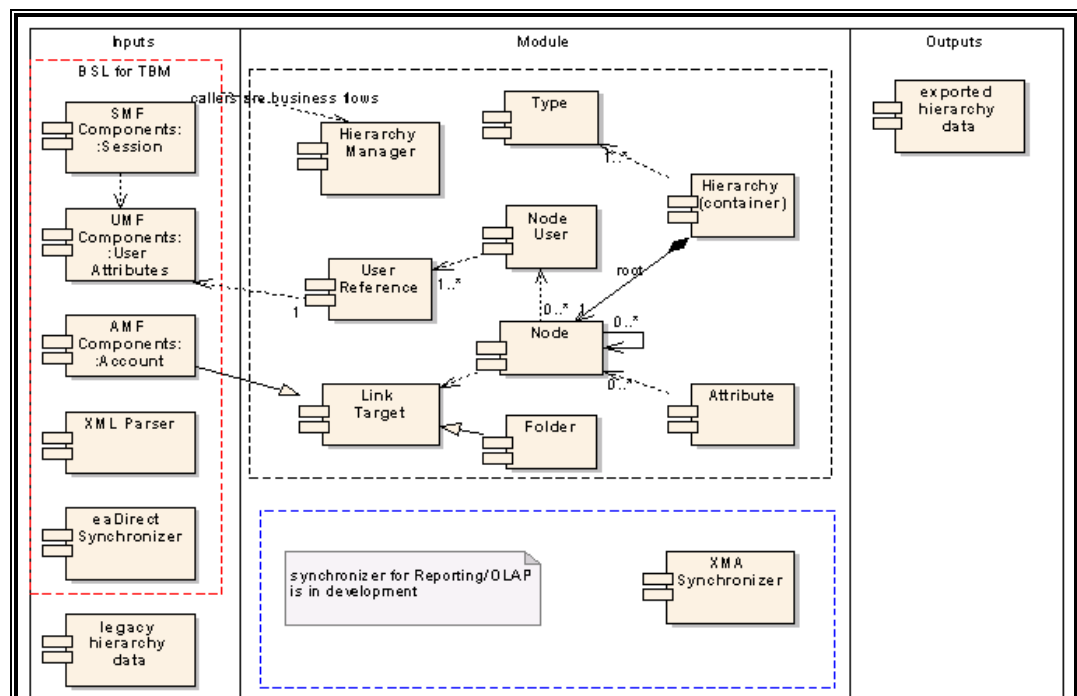
The following example shows how to configure this session bean in your application's `web.xml`:

```
<ejb-ref id="EjbRef_11">
    <ejb-ref-name>ejb/PDFMerger</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.edocs.services.merger.IMergerHome</home>
    <remote>com.edocs.services.merger.IMerger</remote>
</ejb-ref>
```

8

BSL Hierarchy Connector

8.1 Overview



Hierarchy - A system organized in the shape of a pyramid, with each row of objects, sometimes called nodes, linked to objects directly beneath it. A hierarchy contains a root directory at the top of the pyramid and subdirectories below it.

Node - A position in the hierarchy. A node can associate with a link target to indicate what the node represents. Users can be assigned to a node in the hierarchy to determine their access to that node and its subtree. Nodes are organized into parent -child relationships in the hierarchy. Nodes may have any number of name:value attributes.

Link target - A business object that can be associated to a hierarchy node.

URI - universal resource identifier to reference objects that exist in other modules. When a relationship is many-to-many, Hibernate requires that both sides of the relationship exist within the module domain. In order to fully use Hibernate for the hierarchy module, create a reference class within the module to do the many-to-many mapping between source object and reference object (proxy for original target object). In the Hierarchy module, UserReference class is a reference object for the IUser object from the UMF module.

Folder - Predefined link target object inside hierarchy module. Folder's life cycle is managed by its containing hierarchy.

Hierarchy User - A user with permission to view/modify (but not create) a hierarchy. For example CSR or any user in the system as long as they are given the permission.

Permission - A permission allows a user to view or take action on information they have access to, based on the node they are assigned to in the hierarchy. Permissions include view summary, view detail, pay, report, manage hierarchy, assign users, assign permissions, etc.

The Business Services Layer contains a connector package that provides import, synchronization, and hierarchy management for TBM applications. BSL connector employs several sub-systems to create, modify, and synchronize hierarchical information from a variety of sources, including XML files and eStatement DDNs.

Use the BSL connector in concert with other BSL components, the TBM CAM component, the Hierarchy module, Struts action classes, Tiles, and an optional third-party jsp tree tag library to display hierarchies of data in your TBM application. See the J2EE/tbm-app directory for a reference implementation of hierarchies.

See the BSL javadoc delivered with this SDK for detailed implementations of hierarchy import and synchronization interfaces.

The BSL connector relies implicitly on a hierarchy module that is delivered in another TBM jar. All basic hierarchy functionality is provided for by this hierarchy module, which should not need to be customized, as it provides basic hierarchy support: adding, deleting, editing, and searching at the node level. Also programmatic linking of users to nodes provides a means of hierarchy-based access control at this level. A key core class often used is `com.edocs.common.hierarchy.core.HierarchyManager`, which implements `IHierarchyManager` and employs several hierarchy module interfaces and classes to perform the low-level management of hierarchy elements.

See the hierarchy module javadoc delivered with this SDK for detailed descriptions of key hierarchy interfaces and default implementations.

8.2 How to Create and Manage Hierarchies via XML

To manage hierarchies, you can use the following DTD file to validate an XML file that specifies the create, update, and remove operations for hierarchy management:

```

<!-- edited with XMLSPY v2004 rel. 3 U (http://www.xmlspy.com) by Jing Wang
(eDocs Inc) -->

<!--
*****
*****

    *This file cotains the XML format definition that eDocs hierarchy
    module uses for interchanging hierarchical

    * business structures.

    *****
    *****

-->

<!ELEMENT HierarchyList (HierarchyToBeCreated?, HierarchyToBeUpdated?,
HierarchyToBeRemoved?)>

<!ELEMENT HierarchyToBeCreated (Hierarchy*)>
<!ELEMENT HierarchyToBeUpdated (UpdateHierarchy*)>
<!ELEMENT HierarchyToBeRemoved (RemoveHierarchy*)>

<!-- =====
    * Section for creating hierarchy
    =====
-->

<!ELEMENT Hierarchy (Name, Description, NewHierarchyNode)>
<!ATTLIST Hierarchy
    id CDATA #REQUIRED
    companyId CDATA #REQUIRED
    type (billing | business | report) "business"
>

<!ELEMENT NewHierarchyNode ((LinkTargetReference | SimpleNodeFolder), Alias?,
CanBeAccessedBy?, ChildNodeList?)>
<!ELEMENT ChildNodeList (NewHierarchyNode*)>

<!--
<!ELEMENT IdentityNode (ExternalReference | SimpleNodeFolder)>
    Simple Node is the object the keeps information that are hierarchy specific-
->

<!ELEMENT SimpleNodeFolder (Name?, Description, Attribute*)>
<!ATTLIST SimpleNodeFolder
    id CDATA #REQUIRED
>

<!ELEMENT Attribute EMPTY>
<!ATTLIST Attribute
    name CDATA #REQUIRED
    value CDATA #REQUIRED
>

<!ELEMENT LinkTargetReference (Name?, Description?)>
<!ATTLIST LinkTargetReference

```

```

        id CDATA #REQUIRED
        type (BillingAccount | BusinessUnit | CostCenter) "BillingAccount"
        action (Link | Create | Override) "Link"
    >
    <!-- Use this element to describe the list of the users who have granted
    access to the node and all nodes below in the hierarchy. -->
    <!ELEMENT CanBeAccessedBy (UserExternalReference*)>
    <!ELEMENT UserExternalReference EMPTY>
    <!ATTLIST UserExternalReference
        id CDATA #REQUIRED
    >
    <!ELEMENT Alias (#PCDATA)>
    <!ELEMENT Description (#PCDATA)>
    <!ELEMENT Name (#PCDATA)>
    <!ATTLIST HierarchyID
        value CDATA #REQUIRED
    >
    <!-- =====
    *   Section for updating hierarchy
    =====
    -->
    <!ELEMENT UpdateHierarchy (NodesToBeCreated?, NodesToBeUpdated?,
    NodesToBeRemoved?)>
    <!ATTLIST UpdateHierarchy
        id CDATA #REQUIRED
        companyId CDATA #REQUIRED
    >
    <!ELEMENT NodesToBeCreated (NewNode*)>
    <!ELEMENT NewNode (ParentNode, NewHierarchyNode)>
    <!ELEMENT NodesToBeUpdated (UpdateNode*)>
    <!ELEMENT UpdateNode (ParentNode, ChildNode)>
    <!ELEMENT ParentNode EMPTY>
    <!ATTLIST ParentNode
        id CDATA #REQUIRED
        type (BillingAccount | BusinessUnit | CostCenter | Folder)
        "BillingAccount"
    >
    <!ELEMENT ChildNode EMPTY>
    <!ATTLIST ChildNode
        id CDATA #REQUIRED
        type (BillingAccount | BusinessUnit | CostCenter | Folder)
        "BillingAccount"
    >

```

```

<!ELEMENT NodesToBeRemoved (NodeToDelete*)>
<!ELEMENT NodeToDelete EMPTY>
<!ATTLIST NodeToDelete
    id CDATA #REQUIRED
    type (BillingAccount | BusinessUnit | CostCenter | Folder)
    "BillingAccount"
>

<!-- =====
*   Section for removing hierarchy
=====
-->

<!ELEMENT RemoveHierarchy EMPTY>
<!ATTLIST RemoveHierarchy
    id CDATA #REQUIRED
    companyId CDATA #REQUIRED
>

```

Here is a sample hierarchy xml file that follows the above DTD specification:

```

<?xml version="1.0" encoding="UTF-8"?>
<!--Sample XML file generated by XMLSPY v2004 rel. 3 U (http://www.xmlspy.com)-->
<!DOCTYPE HierarchyList SYSTEM "eDocs_Hierarchy_Interchange1.0.dtd">
<HierarchyList>
    <HierarchyToBeCreated>
        <Hierarchy id="100" companyId="COMP-0001" type="billing">
            <Name>eDocs Billing</Name>
            <Description>This is a sample billing
hierarchy</Description>
            <NewHierarchyNode>
                <SimpleNodeFolder id="20001">
                    <Name>HR Accounts</Name>
                    <Description>This folder list all accounts
for HR Department</Description>
                    <Attribute name="Address " value="1 Apple
Hill, Natick, MA"/>
                    <Attribute name="Phone " value="508-652-
8700"/>
                </SimpleNodeFolder>
                <CanBeAccessedBy>
                    <UserExternalReference id="user1"/>
                    <UserExternalReference id="user2"/>
                </CanBeAccessedBy>
                <ChildNodeList>
                    <NewHierarchyNode>

```

```

                                <LinkTargetReference id="acct20001"
type="BillingAccount">
                                <Name>Account #2001 </Name>
                                <Description>This is the
account 20001</Description>
                                </LinkTargetReference>
                                </NewHierarchyNode>
                                <NewHierarchyNode>
                                <LinkTargetReference id="acct20002"
type="BillingAccount">
                                <Description>This is the
account 20002</Description>
                                </LinkTargetReference>
                                </NewHierarchyNode>
                                </ChildNodeList>
                                </NewHierarchyNode>
                                </Hierarchy>
                                <Hierarchy id="200" companyId="COMP-0002" type="billing">
                                <Name>ePays Billing</Name>
                                <Description>This is a sample billing
hierarchy</Description>
                                <NewHierarchyNode>
                                <LinkTargetReference id="c30001" type="CostCenter"/>
                                <CanBeAccessedBy>
                                <UserExternalReference id="user10"/>
                                <UserExternalReference id="user20"/>
                                </CanBeAccessedBy>
                                <ChildNodeList>
                                <NewHierarchyNode>
                                <!--
                                <LinkTargetReference id="c30003"
type="CostCenter"/> -->
                                <SimpleNodeFolder id="40001">
                                <Name>PS Team</Name>
                                <Description>This folder list
all accounts for PS Department</Description>
                                <Attribute name="Company
Address " value="2 Apple Hill, Natick, MA"/>
                                <Attribute name="Company
Phone " value="508-652-8700"/>
                                </SimpleNodeFolder>
                                <Alias>Text</Alias>
                                <CanBeAccessedBy>
                                <UserExternalReference
id="user11"/>
                                <UserExternalReference
id="user22"/>

```

```

        </CanBeAccessedBy>
        <ChildNodeList>
            <NewHierarchyNode>
                <LinkTargetReference
id="acct30003" type="BillingAccount"/>
            </NewHierarchyNode>
            <NewHierarchyNode>
                <LinkTargetReference
id="acct30004" type="BillingAccount"/>
            </NewHierarchyNode>
        </ChildNodeList>
    </NewHierarchyNode>
    <NewHierarchyNode>
        <LinkTargetReference id="acct30001"
type="BillingAccount"/>
    </NewHierarchyNode>
    <NewHierarchyNode>
        <LinkTargetReference id="acct30002"
type="BillingAccount"/>
    </NewHierarchyNode>
</ChildNodeList>
</NewHierarchyNode>
</Hierarchy> -->
</HierarchyToBeCreated>
<HierarchyToBeUpdated>
    <UpdateHierarchy id="200" companyId="COMP0002">
        <NodesToBeCreated>
            <NewNode>
                <ParentNode id="40001" type="Folder"/>
                <NewHierarchyNode>
                    <LinkTargetReference id="acct30005"
type="BillingAccount"/>
                <Alias>Text</Alias>
            </NewHierarchyNode>
        </NewNode>
    </NodesToBeCreated>
    <NodesToBeUpdated>
        <UpdateNode>
            <ParentNode id="c30003" type="CostCenter"/>
            <ChildNode id="acct30001"
type="BillingAccount"/>
        </UpdateNode>
    </NodesToBeUpdated>
    <NodesToBeRemoved>
        <NodeToDelete id="acct30001" type="BillingAccount"/>
    </NodesToBeRemoved>

```

```

        </NodesToBeRemoved>
    </UpdateHierarchy>
</HierarchyToBeUpdated>
<HierarchyToBeRemoved>
    <RemoveHierarchy id="100" companyId="COMP-0001"/>
    <RemoveHierarchy id="200" companyId="COMP-0002"/>
</HierarchyToBeRemoved>
</HierarchyList>

```

To manage the hierarchies specified in the above XML, use or extend connector/tasks/HierarchyImporter.java and register it in a job in the Command Center. See the chapter, *Custom Jobs*, for details on how to do this.

HierarchyImporter loads hierarchies from XML files. HierarchyImporter calls com.edocs.common.bsl.connector.HierarchyXMLParser to read the file. See the BSL connector/tasks javadoc for implementation details.

8.3 How to Synchronize Hierarchies with eStatement Indexer Data

Another way to create hierarchies for the TBM application is to obtain the information from the eStatement Indexer database table using a Command Center job. Use or extend connector/tasks/BillingDataSynchronizer.java and register it in a job in the Command Center.

BillingDataSynchronizer synchronizes pre-existing hierarchies with eStatement indexer data. BillingDataSynchronizer calls BillingHierarchyLoader(aBillingHierSyncHandler) to perform the synchronization.

The classpath resource, hierarchy.cfg.xml, is used to configure the hierarchy synchronization handler bean. See the default example delivered with TBM in bsl/config:

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">
<beans>
    <!-- Billing hierarchy handler bean -->
    <bean id="billingHierarchyHandler"
class="com.edocs.common.bsl.connector.BillingHierSyncHandler"
singleton="false">
        <property name="serviceBuilder">
            <ref local="bslServiceBuilder"/>
        </property>
        <property name="billingHierarchyName">
            <value>BILLING</value>
        </property>
    </bean>

```

```

        </property>
        <property name="defaultCompanyName">
            <value>__default__</value>
        </property>
        <property name="companyColumnName">
            <value>CustName</value>
        </property>
    </bean>

    <bean id="bslServiceBuilder"
class="com.edocs.common.bsl.connector.BusinessServiceBuilder"
singleton="true">
        <property name="userId">
            <value>admin</value>
        </property>
        <property name="userPassword">
            <value>edocs</value>
        </property>
    </bean>

    <bean id="billingHierarchyBuilderProxy"
class="org.springframework.aop.framework.ProxyFactoryBean">
        <property name="target">
            <ref local="billingHierarchyHandler"/>
        </property>
        <!-- <property
name="interceptorNames"><value>afterReturningInterceptor</value></property> -
-->
        <property name="interceptorNames">
            <value>billingSyncAdvisor</value>
        </property>
    </bean>

    <bean id="billingSyncAdvisor"
class="org.springframework.aop.support.DefaultPointcutAdvisor">
        <property name="pointcut">
            <bean
class="com.edocs.common.hierarchy.connector.BillingHierSyncPointcut"/>
        </property>
        <property name="advice">
            <ref local="afterReturningInterceptor"/>
        </property>
    </bean>

    <!-- Replace class name with your own interceptor class name here-->
    <bean id="afterReturningInterceptor"
class="com.edocs.common.bsl.connector.tasks.AccountStatementInterceptor"/>
</beans>

```

The synchronizer may not pick up all the indexed data that is available from eStatement by default. To extract more data or do additional processing after the synchronization handler processes each row in the indexer table, create an interceptor and register it in hierarchy.cfg.xml.

In BSL's connector/tasks package, *AccountStatementInterceptor* implements *org.springframework.aop.AfterReturningAdvice*. But you can also create an interceptor by extending *com.edocs.common.hierarchy.api.BillingHierAfterReturningInterceptor* and overriding *afterReturningHook()*:

```
/**
 * Override this method to do any additional processing.
 * Called every time a row in the indexer table is processed.
 * Each row represents either a parent account or a nested
 * sub-account or Mobile Telephone Number (MTN). Uses Z-Context
 * field to determine how to sort rows into unresolved and accounts
 * hashtables.
 *
 * @param handler handler object that synchronizes billing hierarchy with
 * indexed billing information.
 * @param zPrimary Z_PRIMARY field value in the current row
 * @param zContext Z_CONTEXT field value in the current row
 * @param rowData all other field vlaues in the current row.
 * Each field is indexed by the column name.
 */
protected void afterReturningHook(IBillingHierSyncHandler handler, String
zPrimary, String zContext, HashMap rowData) throws Exception {
    String statementDate = (String)rowData.get("StatementDate");

    String[] amount = {(String)rowData.get("TotalAmtDueAmt"),
    (String)rowData.get("MTNCurrChgs"), (String)rowData.get("MTNCurrChgsCR")};

    log.debug("AccountStatementInterceptor.afterReturningHook(): Z_PRIMARY =
" + zPrimary + "; Z_CONTEXT=" + zContext + "; StatementDate=" + statementDate +
        "; TotalAmtDueAmt=" + amount[0] + "; TotalMtnAmt="+amount[1] + ";
TotalMtnAmt="+amount[2]);

    if (zContext==null || zContext.trim().length()<1) {
        // Top-level Account
        _store(zPrimary, statementDate, _resolveAmount(amount));

    } else {
        // Nested Account or MTN
        if (statementDate!=null) {
            _store(zPrimary, statementDate, _resolveAmount(amount));
        }
    }
}
```

```

    } else {
        BillingStatement statement =
            (BillingStatement)accounts.get(zContext);
        if (statement==null) {
            // Add MTN to the unresolved pile
            Vector unMtns = (Vector)unresolved.get(zContext);
            if (unMtns==null) {
                unMtns = new Vector();
                unresolved.put(zContext, unMtns);
            }

            // add to unresolved list
            String[] data = {zPrimary, _resolveAmount(amount)};
            unMtns.add(data);

        } else {
            // Parent Account found
            _store(zPrimary, statement.getStatementDate(),
                _resolveAmount(amount));
        }
    }
}

```

AfterReturningHook() is called after each row in the indexer table is processed by the synchronizer. The rowData hashMap contains all the values in the record, keyed by the column name. You can also get table name, ivn, and ddn through the handler class which is passed in as the first argument.

Be sure to add the interceptor class to the bean definition in *hierarchy.cfg.xml* as follows:

```

<bean id="afterReturningInterceptor"
    class="com.edocs.common.bsl.connector.tasks.AccountStatementInterceptor"/>

```

Place hierarchy.cfg.xml anywhere in the deployment classpath. In order to compile the interceptor class, the following jars should be deployed in the lib directory:

- Aopalliance-1.0.jar
- Spring-1.0.2.jar

8.4 Hierarchy APIs

See the BSL connector javadocs for implementation details about high-level hierarchy management. Refer to the hierarchy module javadocs for implementation details about low-level hierarchy structures.

Customization of low-level hierarchy management may never be required, as most of the work needed to manage hierarchies at this level is fully implemented as follows. The package to look for in the hierarchy module jar is `com.edocs.common.hierarchy.api`.

8.4.1 Creating Hierarchies Programmatically

To create hierarchy from scratch:

```

IHierarchyManager hMgr = new HierarchyManager()
String companyId = new String("COMP-1");
String hierName = new String ("BILLING");
IHierarchyType hierType = Hierarchy.BUSINESS;
IHierarchy hier = hMgr.createHierarchy(companyId, hierName, hierType);
...

```

To create hierarchy from an existing hierarchy:

```

IHierarchyNode rootNode = hier.getRoot();
IHierarchy anotherHier = createHierarchyFromNode(root);

```

8.4.2 Adding Entities to a Hierarchy

Any Business Objects that implement `IHierarchyLinkTarget` can be added into Hierarchy:

```

public interface IBusinessObject {
    public String getExternalID();
    public String getIdentityURI();
    public String getDisplayName();
    public String getTypeCode();
}

public interface IHierarchyLinkTarget extends IBusinessObject
    boolean isContainer();

```

To add a link target to Hierarchy:

```

IHierarchyLinkTarget linkAcct = bs.findAccount("acc100");
IHierarchyNode acctNode = hMgr.addLinkTarget(rootNode ,linkAcct);

```

To add a folder to Hierarchy:

```

IHierarchyFolder hFolder = new HierarchyFolder("HR", "Human Resource",
                                                "This is HR folder");
hFolder.addAttribute(new Attribute("Phone: ", "508-123-8700"));
IHierarchyNode fNode = hMgr.addFolder hMgr.addFolder(rootNode, hFolder);
}

```

To add a node to Hierarchy:

```
hMgr.addNode(fNode, acctNode);
```

8.4.3 Finding Hierarchies, Nodes, and Folders

Developers decide how much data to load: Lazy (metadata only) or Bulk Loads.

To load Hierarchy MetaData (container) only:

```
getAllHierarchyMetaData()
getAllHierarchyMetaData(hierarchyType);
getAllHierarchyMetaDataForUser(userName);
getAllHierarchyMetaDataForUser(hierarchyType, userName);
```

To load Hierarchy including its tree structure:

```
getHierarchy("COMP-1", "BILLING", true);
Returns the whole tree
getHierarchyForUser("COMP-1", "BILLING", "jwang", true);
Returns a list of root nodes that the given user has access to
```

To load Hierarchy meta data and non-leaf node only:

```
getHierarchyStructure("COMP-2", "BUSINESS")
```

To locate a node in Hierarchy:

```
IHierarchyNode findNodeByIdentityURI(IHierarchy hier, String uri)

IHierarchyNode[] findObjNodeByIdentityId(String identityId)
IHierarchyNode[] findObjNodeByIdentityId(IHierarchy hier, String identityId)
IHierarchyNode[] findObjNodeByIdentityId(IHierarchyNode hierNode, String
identityId)

IHierarchyNode[] findObjectsNodeByURI(String uri)
IHierarchyNode findObjNodeByURI(IHierarchy hier, String uri)
IHierarchyNode findObjNodeByURI(IHierarchyNode hierNode, String uri)
```

To locate a folder in Hierarchy:

```
■ IHierarchyNode [] findFolderNode(IHierarchy hier, String name)
■ IHierarchyNode[] findFolderNode(IHierarchNode node, String name)
```

8.4.4 Updating Nodes

To update a node:

```
updateFolder(IHierarchy hierarchy, IHierarchyFolder folder);
updateNode(IHierarchyNode hNode);
```

To move node to a different parent:

```
moveNodeTo(IHierarchyNode node, IHierarchyNode destinationNode)
```

To delete a node:

```
deleteNode(IHierarchyNode nodeToBeDeleted)
```

8.4.5 Giving User Access to Nodes

To give user access to a node:

```
addUserAccess(IHierarchyNode hNode, String userId)
addUserAccess(IHierarchy hier, IHierarchyLinkTarget target,
              String userId)
```

To remove user access from a node:

```
removeUserAccess(IHierarchyNode hNode, String userId)
removeUserAccess(IHierarchy hier, IHierarchyLinkTarget target,
              String userId)
```

Permission types come from user object.

8.5 How to Implement Hierarchy-Based Access Control (HBAC)

HBAC provides another level of security for your TBM application. When a user enrolls, a security profile registers one or more roles for the user. The SAF sub-system tests the logged-in user's permission to access key parts of the application. When a user is assigned to specific nodes in a business structure hierarchy, the access of the user is further limited via HBAC.

To implement HBAC, you can use the *CanBeAccessedBy* element in an XML file to add user access to specific nodes in the hierarchy as follows:

```
...

<CanBeAccessedBy>
    <UserExternalReference id="user1"/>
    <UserExternalReference id="user2"/>
</CanBeAccessedBy>
...
```

You can do this programmatically by extending the BSL core/BusinessServices class or re-implementing bsl/api/IBusinessServices methods to assign users to specific nodes according to your needs. You might constrain hierarchy access by role as well. Or the logged-in user's account list might be filtered through CAM according to account-level pattern matching. The flexibility provided by TBM makes the final solution entirely up to you.

See the CAM and BSL javadocs for default implementations available. Also see the J2EE/app-b2b source code for a reference implementation of HBAC.

8.6 How to Search for and Find Objects Within Hierarchies

Implement BSL's api/IBusinessServices.java or use BSL's core/BusinessServices class, which implements IBusinessServices, to search existing hierarchies for pertinent information.

Using IBusinessServices methods you can locate and return whole hierarchies or just the metadata for hierarchies. You can also search for objects within hierarchies. There is a method for every search that is based on a specific filter parameter.

See the BSL/api javadoc for implementation details.

9.1 Overview

The Business Services Layer contains interfaces and default implementations for user profile management, security profile management, and company profile management. The packages for these components are as follows:

- User profile management – bsl/umf
- Security profile management – bsl/authentication
- Company profile management – bsl/cmf

9.2 How to Manage the User Profile

When a user enrolls, a user profile is created and stored in the database. The user profile may be programmatically managed through the BSL via implementations of `IUserProfileManger` and other interfaces in the umf package. The BSL controller, `IBusinessServices`, calls these umf classes to access user profile information. See the BSL/umf javadoc for default implementations that may be extended to customize user profile management.

9.3 How to Manage the Security Profile

When a user enrolls, a security profile is created and stored in the database. The security profile is an important aspect of SAF and role-based access control. The security profile may be programmatically managed through the BSL via implementations of `ISecurityProfileManger` and other interfaces in the authentication package. The BSL controller, `IBusinessServices`, calls these authentication classes to access security profile information. See the BSL/authentication javadoc for default implementations that may be extended to customize security profile management.

9.4 How to Manage the Company and Company User Profiles

When a user enrolls, a company and company user profile may be created and stored in the database. The company profile may be programmatically managed through the BSL via implementations of `ICompanyProfileManger` and other interfaces in the `cmf` package. The BSL controller, `IBusinessServices`, calls these `cmf` classes to access company and company user profile information. See the BSL/`cmf` javadoc for default implementations that may be extended to customize company profile management.

10.1 Notification Setup UI

E-mail notification setup tiles (*.jsp) are delivered in a WAR file in the tbm-b2b-app EAR. The package location for these tiles is webroot/profile. The tiles-defs.xml and struts-config.xml files shown in tables below are delivered in webroot/WEB-INF.

Notifications are configured under the Profiles tab in the TBM application UI. See webroot/WEB-INF/tiles-defs.xml to configure the add, edit, and delete email tiles delivered with TBM:

...

```
<!-- Profile Sub Tab List -->
    <definition name=".tabs.profile" extends=".template.tabs">
        <putList name="subtabList">
            <item value="Personal Profile"
link="/profile/personal_profile.do"
classtype="com.edocs.application.tbm.core.menus.EdocsBaseMenuItem"/>
            <item value="Notifications"
link="/profile/notifications.do"
classtype="com.edocs.application.tbm.core.menus.EdocsBaseMenuItem"/>
            <item value="Personal Address Book"
link="/profile/personal_address_book.do"
classtype="com.edocs.application.tbm.core.menus.EdocsBaseMenuItem"/>
        </putList>
    </definition>
...
<!--Notification tiles -->
    <definition name=".main.profile.notifications" extends=".template.main">
        <put name="sectionHeader" value=".template.sectionHeader"
type="definition"/>
        <put name="sectionHeadline" value="Profile"/>
        <put name="subSectionHeadLine" value="Notifications"/>
        <put name="tabs" value=".tabs.profile" type="definition"/>
        <put name="tab" value="Profile"/>
        <put name="subtab" value="Notifications"/>
        <put name="leftSideNav" value="/_includes/side_navbar.jsp"
type="page"/>
        <put name="module1" value=".module.profile.notifications"
type="definition"/>
```

```

</definition>

<definition name=".module.profile.notifications"
extends=".template.module">
    <put name="form" value="/profile/notifications.jsp" type="page"/>
</definition>

<definition name=".main.profile.notifications.add"
extends=".main.profile.notifications">
    <put name="module1" value=".module.profile.notifications.add"
type="definition"/>
</definition>

<definition name=".module.profile.notifications.add"
extends=".template.module">
    <put name="form" value="/profile/add_email.jsp" type="page"/>
</definition>

<definition name=".main.profile.notifications.edit"
extends=".main.profile.notifications">
    <put name="module1" value=".module.profile.notifications.edit"
type="definition"/>
</definition>

<definition name=".module.profile.notifications.edit"
extends=".template.module">
    <put name="form" value="/profile/edit_email.jsp" type="page"/>
</definition>

<definition name=".main.profile.notifications.delete"
extends=".main.profile.notifications">
    <put name="module1" value=".module.profile.notifications.delete"
type="definition"/>
</definition>

<definition name=".module.profile.notifications.delete"
extends=".template.module">
    <put name="form" value="/profile/delete_email.jsp" type="page"/>
</definition>

```

When the notification sub-tab is clicked, the Struts action, /profile/notifications.do, defined in struts-config.xml below calls com.edocs.application.tbm.b2b.users.actions.GetAllUserNotificationEmailsAction. On success the action forwards to the tile, .main.profile.notifications, which loads webroot/profile/notifications.jsp into the module1 location (see the previous table). The Struts actions for adding, editing, and deleting email are defined in struts-config.xml:

```

<action path="/profile/notifications"
type="com.edocs.application.tbm.b2b.users.actions.GetAllUserNotificationEmailsAction" scope="request">
    <forward name="success"
path=".main.profile.notifications"/>
</action>

<action path="/profile/notifications/add/show"
forward=".main.profile.notifications.add">

```

```

        </action>

        <action path="/profile/notifications/add/perform"
type="com.edocs.application.tbm.b2b.users.actions.UpdateUserNotificationEmail
Action" scope="request" name="emailForm" parameter="add" validate="true"
input="addEmail">

            <forward name="success"
path="/profile/notifications.do"/>

        </action>

        <action path="/profile/notifications/edit/show"
type="com.edocs.application.tbm.b2b.users.actions.GetUserNotificationEmailAct
ion" scope="request" name="editEmailForm" validate="false">

            <forward name="success"
path=".main.profile.notifications.edit"/>

        </action>

        <action path="/profile/notifications/edit/perform"
type="com.edocs.application.tbm.b2b.users.actions.UpdateUserNotificationEmail
Action" scope="request" name="editEmailForm" parameter="edit" validate="true"
input="editEmail">

            <forward name="success"
path="/profile/notifications.do"/>

        </action>

        <action path="/profile/notifications/delete/show"
type="com.edocs.application.tbm.b2b.users.actions.GetUserNotificationEmailAct
ion" scope="request" name="emailForm" validate="false">

            <forward name="success"
path=".main.profile.notifications.delete"/>

        </action>

        <action path="/profile/notifications/delete/perform"
type="com.edocs.application.tbm.b2b.users.actions.UpdateUserNotificationEmail
Action" scope="request" name="emailForm" parameter="delete" validate="true"
input="deleteEmail">

            <forward name="success"
path="/profile/notifications.do"/>

        </action>

```

10.2 Notification Setup Actions

The Struts actions for TBM Notifications are delivered with the `tbm-b2b-app` WAR in `com.edocs.application.tbm.b2b.users.actions`:

- `GetAllUserNotificationEmailsAction.java`
- `GetUserNotificationEmailAction.java`
- `UpdateUserNotificationEmailAction.java`

A code snippet from a `doAction()` method for the update action demonstrates how the Business Services Layer is employed to get the current user's email notification configuration and generate new email notifications. The BSL components, `util/IEmail` and `util/DefaultEmail`, and `umf/api/IUser` are shown in use below:

```

public ActionForward doAction(ActionMapping mapping,
                              ActionForm form,
                              HttpServletRequest request,
                              HttpServletResponse response)
    throws Exception {
    ...
    EmailForm f = (EmailForm)form;
    String uid = SessionUtils.getUserId(request);

    IBusinessServices ibs = new BusinessServices();
    log.debug("Instantiated business services - " + ibs);

    try {
        IUser u = ibs.findUserByUID(uid);
        Set emails = u.getNotificationEmails();

        try {
            if (f.getAction().equalsIgnoreCase(EmailForm.ACTION_ADD)) {
                log.debug("ACTION: ADD");
                add(emails, f);
            } else if (f.getAction().equalsIgnoreCase(EmailForm.ACTION_EDIT)) {
                log.debug("ACTION: EDIT");
                edit(emails, f);
            } else if (f.getAction().equalsIgnoreCase(EmailForm.ACTION_DELETE)) {
                log.debug("ACTION: DELETE");
                remove(emails, f);
            }
        } catch (Exception e) {
            // email to update not found - throw error
            log.error("Exception occurred: " + e.getMessage());
            e.printStackTrace();
            errors = new ActionErrors();
            errors.add(ActionErrors.GLOBAL_ERROR, new
                ActionError(e.getMessage()));
            saveErrors(request, errors);
            return (mapping.findForward(RIConstants.ERROR));
        }

        // Update user
        ibs.updateUser(u);
    }
}

```

```

        } catch (Exception e) {
            //new
...
        ActionError("com.edocs.error.customer.already.enrolled"));
        log.error("Exception occurred: " + e.getMessage(), e);
        errors = new ActionErrors();
        errors.add(ActionErrors.GLOBAL_ERROR, new
        ActionError(RIErrorMessage.EXCEPTION_OCCURRED));
        saveErrors(request, errors);
        return (mapping.findForward(RIConstants.ERROR));
    }

    private void add(Set emails, EmailForm f) throws Exception {
        IEmail email = null;
        // verify if email already exist
        Iterator it = emails.iterator();
        while (it.hasNext()) {
            IEmail e = (IEmail)it.next();
            if (e.getEmailAddress().equalsIgnoreCase(f.getEmailAddress())) {
                email = e;
                break;
            }
        }
        if (email!=null) {
            // email already exist - throw error
            throw new Exception("User already has email:
email="+f.getEmailAddress());
        }
        email = new DefaultEmail(f.getEmailAddress());
        emails.add(email);
        if (f.isPrimary()) {
            resetPrimary(emails);
            email.setPrimary(true);
        }
    }
...

```

The Struts ValidatorForm form for all TBM Notification actions is com.edocs.application.tbm.b2b.users.forms.EmailForm.

10.3 Messenger Component

Once email notifications have been set up using the TBM profile UI, configure back-end e-mail delivery in app-config.properties:

```
# SMTP server for email notification for enrollment
#mail.smtp.host=10.1.33.51

## Messenger Properties:
## dispatcherSleepPeriod and hangingTimeout are measured in seconds
##
mail.host=exchangeus.edocs.com
#   mail.user=
#   mail.from=
#   mail.store.protocol=
mail.transport.protocol=SMTP
#   mail.debug=true
mail.queue.threadMax=10
mail.queue.elementsPerThread=30
mail.queue.dispatcherSleepPeriod=5
mail.queue.hangingTimeout=10
mail.queue.storagedDirectory=mailqueue
mail.xml.templateFilename=templates.xml
```

Configure text messages in /webroot/WEB-INF/messenger-config.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<messenger-config>
  <defaults>
    <from>customerservice@vzw.com</from>
  </defaults>
  <message-type id="simple-test">
    <subject>Simple Test</subject>
    <from/>
    <text><![CDATA[
This is a message with no dynamic data.
]]></text>
  </message-type>
  <message-type id="Forgot.Password">
    <subject>Your temporary password at www.vzw.com</subject>
    <from/>
```

```

        <text><![CDATA[
Your temporary password for %uid.toString()% for logging into
https://www.vzw.com/tbmb is %password.toString()%. Please log into the system
and change your password when prompted.

]]></text>

</message-type>

<message-type id="Enroll.Complete">

    <subject>Thank you for enrolling with
http://www.vzw.com/tbmb</subject>

    <from/>

    <text><![CDATA[

        Dear %uid%,

        Welcome to Telekom Service, your temporary password is
        %password.toString()%. Now you can enjoy more control in managing and paying your
        bills. Why wait! Log in now into the system and you are advise to change your
        password for security reason.

        To log in please go to https://www.vzw.com/tbmb

    ]]></text>

</message-type>

</messenger-config>

```

The messenger component merges the message template above with runtime information to create an email message. The messenger then calls the gateway, configured in app-config.properties, to send the email message. For most TBM custom applications, email is sent during custom job or payment processing.

See the javadocs delivered for the following TBM messenger components:

- com.edocs.messenger.api.AMessage
- com.edocs.messenger.api.IParameterBuilder
- com.edocs.messenger.exception.MessengerException
- com.edocs.messenger.exception.MessengerInitException
- com.edocs.messenger.exception.MessengerQueueException
- com.edocs.messenger.exception.MessengerTemplateException
- com.edocs.messenger.exception.MessengerTransportException
- com.edocs.messenger.extensions.GroupTransporter
- com.edocs.messenger.extensions.InternalMessage
- com.edocs.messenger.extensions.QueueTransporter
- com.edocs.messenger.extensions.TrueTransporter
- com.edocs.messenger.extensions.XSLTemplate

```

com.edocs.messenger.extensionsapi.ATemplate
com.edocs.messenger.extensionsapi.ATransporter
com.edocs.messenger.group.GroupTransportTable
com.edocs.messenger.queue.QueueDispatcher
com.edocs.messenger.queue.QueueDispatchHandler
com.edocs.messenger.queue.QueueTransportTable
com.edocs.messenger.Example1Message
com.edocs.messenger.Example2Message
com.edocs.messenger.Messenger
com.edocs.messenger.SimpleMessage

```

10.4 How to Write Custom Jobs for Email Notification

Notifications and alerts can be sent during custom job processing. Read Chapter 12, Custom Jobs. Then implement `ICustomJob` and register it as a custom job task. This code snippet demonstrates the flow of control:

```

..
public class PaymentStatusEmailTask implements ICustomJob {
    static final boolean DEBUG = Boolean.getBoolean("com.edocs.ps.debug");

    private String ddn;
    private String additionalDdnList;

    public boolean run() throws Exception {

        JaxorContext context = null;
        Connection connection = null;

        try {
            connection = DataSourceFactory.newDataSource().getConnection();
            if (connection == null) {
                throw new Exception("Could not get the Database connection using
DataSourceFactory");
            }

            context = new JaxorContextImpl(connection);

```

```

        if (context == null) {
            throw new Exception("Could not create the Jaxor context using
JaxorContextImpl");
        }

        Emailer emailer = EmailerFactory.getInstance();
        TuftsEmailAddressResolver resolver =
TuftsEmailAddressResolverFactory.getInstance();
        List statusList = getStatusList(resolver);
        String ddnReferenceList = DdnHelper.getReferenceList(ddn,
additionalDdnList, context);
        Map achErrorLookup = getAchErrorLookup(context);

        for (Iterator iterator = statusList.iterator(); iterator.hasNext();)
        {

            PaymentStatus status = (PaymentStatus) iterator.next();
            if (DEBUG) System.out.println("status=" + status);

            PaymentStatusEmailCommand command = new
PaymentStatusEmailCommand(status, ddnReferenceList, resolver, emailer,
achErrorLookup, context);
            command.execute();

        }
        return true;
    }
    finally {
        if (connection != null)
            connection.close();
        if (context != null)
            context.end();
    }
}
}
...

```

The run() method above uses PaymentStatusEmailCommand.execute() to execute the message. Here is a code snippet from PaymentStatusEmailCommmand.execute() that demonstrates the flow of control from this point:

```

public class PaymentStatusEmailCommand {
    private static final boolean DEBUG =
Boolean.getBoolean("com.edocs.ps.debug");
    private static final String UNKNOWN_ACH_ERROR_DESCRIPTION = "Unknown";
    private static final String INVOICE_NUMBER_PROPERTY_NAME = "INVOICE-NUMBER";

```

```

private PaymentStatus status;
private EmailAddressResolver resolver;
private Messenger messenger;
private Map achErrorLookup;
private String ddnReferenceList;
private JaxorContext context;

public PaymentStatusEmailCommand(PaymentStatus status, String
ddnReferenceList, EmailAddressResolver resolver, Emailer emailer, Map
achErrorLookup, JaxorContext context) {
    this.status = status;
    this.ddnReferenceList = ddnReferenceList;
    this.resolver = resolver;
    this.emailer = emailer;
    this.achErrorLookup = achErrorLookup;
    this.context = context;
}

public void execute() throws Exception {
    try {
        ...
        Map paymentsByAddress = new HashMap();

        Collection payments = findPayments(baseFinder, status,
ddnReferenceList);

        // Get email addresses by group number
        Map addressesByGroup = getAddresses(groupNumbers,
status.getPermissionId());
        if (addressesByGroup == null || addressesByGroup.size() == 0) {

            // Group the payments by email address
            for (Iterator i = groupNumbers.iterator(); i.hasNext();) {
                String groupNumber = (String) i.next();
                Collection groupAddresses = (Collection)
addressesByGroup.get(groupNumber);
                if (groupAddresses == null) // sanity check, some times this will
never happen
                    continue;

                for (Iterator j = groupAddresses.iterator(); j.hasNext();) {

```

```

        String address = (String) j.next();
        if (DEBUG) System.out.println("Email address for the group:"
+ address);

        getBucket(paymentsByAddress,
address).addAll(getBucket(paymentsByGroup, groupNumber));
    }
}

// Send out the emails
for (Iterator i = paymentsByAddress.entrySet().iterator();
i.hasNext();) {
    Map.Entry entry = (Map.Entry) i.next();
    String address = (String) entry.getKey();
    List addressPayments = (List) entry.getValue();
    if (DEBUG) {
        System.out.println("Email Address :" + address);
        System.out.println("Number of Payments :" +
addressPayments.size());
    }
    if (sendMessage(address, addressPayments)) {
        for (Iterator pmtIter = addressPayments.iterator();
pmtIter.hasNext();) {
            CheckPaymentEntity checkEntity = (CheckPaymentEntity)
pmtIter.next();

            turnOffNotificationFlag(checkEntity);
        }
    }
}

context.getTransaction().commit();
if (DEBUG) System.out.println("Transaction committed");

}
catch (Exception e) {
    System.out.println("Error Occured: " + e);
    context.getTransaction().rollback(e);
    throw (e);
}
finally {
}
return;
}

private boolean sendMessage(String address, List payments) throws Exception
{
    try {
        List formattedPayments = formatPayments(payments);

```

```

        PaymentStatusEmailRequest request = new
PaymentStatusEmailRequest(status.getMessageType(), address, formattedPayments);
        messenger.send(request);

        return true;
    }

    catch (EmailerException e) {
        //throw (e);
        System.out.println("Error in sending emails for the address :" +
address);
        System.out.println("Error is:" + e);
        return false;
    }

    catch (Exception e) {
        if (DEBUG) System.out.println("Critical error is occurred while
sending email");
        throw (e);
    }
}

```

So a group of payments can trigger a group of email messages to notify payees that payments were successfully made. In the example above, addresses are resolved through a special payment email resolver EJB. But these addresses can also come from the notification profile that is stored in the database through the TBM Notification UI.

11.1 Address Book Overview

TBM supports personal and corporate address books. These address books are displayed in TBM LDE views, so each has its own DDN. Configure address book DDN's in app-config.properties as follows:

```
# PAB DDN
DDN.PersonalAddressBook=PAB

# CAB DDN
DDN.CorporateAddressBook=CAB
```

See the javadocs delivered for the LDEToolkit component and Chapter 3, Statementing and Content Access for details about TBM Views and Live Data Extraction for Views.

11.2 Personal Address Books

TBM supports personal address books and follows a standard application flow architecture involving Struts and Tiles. Inspect the WAR file delivered in the TBM application EAR to see the configuration and jsp files described below.

The top-level TBM UI involves templates defined in webroot/WEB-INF/tiles-defs.xml and struts-config.xml. TBM delivers a tabbed interface placing a personal address book sub-tab under a profile tab. This is configured in tiles-defs.xml as follows:

```
<tiles-definitions>
<definition name=".template.main" path="/_templates/main_template.jsp">
    <put name="title" value="Welcome to Telco Wireless - We work
for you " type="string"/>
    <put name="header" value="/_includes/header.jsp" type="page"/>
    <put name="sectionHeader" value="/_includes/blank_page.jsp"
type="page"/>
    <put name="sectionHeadline" value="" type="string"/>
    <put name="subSectionHeadLine" value="" type="string"/>
</definition>
</tiles-definitions>
```

```

        <put name="selectDisplayMenu" value="/_includes/blank_page.jsp"
type="page"/>
        <put name="tabs" value=".template.tabs" type="definition"/>
        <put name="tab" value="Account Sum"/>
        <put name="subtab" value="" type="string"/>
        <put name="action" value="/_includes/blank_page.jsp"
type="page"/>
        <put name="leftSideNav" value="/_includes/blank_page.jsp"
type="page"/>
        <put name="summary" value="/_includes/blank_page.jsp"
type="page"/>
        <put name="module1" value="/_includes/blank_page.jsp"
type="page"/>
        <put name="module2" value="/_includes/blank_page.jsp"
type="page"/>
        <put name="module3" value="/_includes/blank_page.jsp"
type="page"/>
        <put name="module4" value="/_includes/blank_page.jsp"
type="page"/>
        <put name="module5" value="/_includes/blank_page.jsp"
type="page"/>
        <put name="module6" value="/_includes/blank_page.jsp"
type="page"/>
        <put name="module7" value="/_includes/blank_page.jsp"
type="page"/>
        <put name="view" value="/_includes/blank_page.jsp"
type="page"/>
        <put name="sidebar" value="sidebar.simple" type="definition"/>
        <put name="footer" value="/_includes/footer.jsp" type="page"/>
    </definition>
<definition name=".template.sectionHeader"
path="/_includes/section_header.jsp"/>
    <!-- Top Navigation Definitions -->
    <definition name=".template.tabs" path="/_templates/tab_template.jsp">
        <putList name="tabList">
            <item value="Overview"
link="/hierarchy/main/dashboard/show.do"
classtype="com.edocs.application.tbm.core.menus.EdocsBaseMenuItem"/>
            ...
        </putList>
        <putList name="subtabList"/>
    </definition>
    ...
    <!-- Profile Sub Tab List -->
    <definition name=".tabs.profile" extends=".template.tabs">
        <putList name="subtabList">

```

```

        <item value="Personal Profile"
link="/profile/personal_profile.do"
classtype="com.edocs.application.tbm.core.menus.EdocsBaseMenuItem"/>
        <item value="Notifications"
link="/profile/notifications.do"
classtype="com.edocs.application.tbm.core.menus.EdocsBaseMenuItem"/>
        <item value="Personal Address Book"
link="/profile/personal_address_book.do"
classtype="com.edocs.application.tbm.core.menus.EdocsBaseMenuItem"/>
    </putList>
</definition>

```

The profile sub tab list above indicates that when the Personal Address Book subtab is selected, navigation proceeds to the Struts action, */profile/personal_address_book*.

Struts-config.xml defines this action as follows:

```

<action path="/profile/personal_address_book"
forward=".main.profile.personalAddressBook">
    </action>

```

The action forwards to another tile that is defined in tiles-defs.xml as follows:

```

<!-- Personal Address Book -->
    <definition name=".main.profile.personalAddressBook"
extends=".template.main">
        <put name="sectionHeader" value=".template.sectionHeader"
type="definition"/>
        <put name="sectionHeadline" value="Profile"/>
        <put name="subSectionHeadLine" value="Personal Address Book"/>
        <put name="tabs" value=".tabs.profile" type="definition"/>
        <put name="tab" value="Profile"/>
        <put name="subtab" value="Personal Address Book"/>
        <put name="leftSideNav" value="/_includes/side_navbar.jsp"
type="page"/>
        <put name="module1" value="/personaladdressbook.do"
type="definition"/>
        <put name="module3" value="/addressbook/pab-addlink.jsp"
type="page"/>
    </definition>

```

Personal address book modules above (module1 and module3) are configured in webroot/WEB-INF/tiles-defs-addressbook.xml and struts-config-addressbook.

module3 above places an add link on the page. The link points to the Struts action class, *com.edocs.application.tbm.b2b.actions.PersonalShowAddressAction*, in struts-config-addressbook:

```

<!-- Show the PAB AddNew Page -->

```

```

        <action path="/addressbook/pab-addnew/show"
type="com.edocs.application.tbm.b2b.actions.PersonalShowAddressAction"
name="addressBookForm" scope="request" parameter=".main.addressbook.pab.addnew"
validate="false">

        <!-- <set-property property="checkLogin" value="false" /> -
->

        <forward name="success"
path=".main.addressbook.pab.addnew"/>

    </action>

```

The action class forwards to .main.addressbook.pab.addnew which pops up another tile, .module.addressbook.pab.addnew, both defined in tiles-defs-addressbook:

```

<definition name=".main.addressbook.pab.addnew" extends=".template.popup">

    <put name="module1" value=".module.addressbook.pab.addnew"/>

</definition>

<definition name=".module.addressbook.pab.addnew"
extends=".template.module">

    <put name="form" value="/addressbook/pab-addnew.jsp" type="page"/>

    <put name="activeModule" value="addressbook"/>

    <putList name="moduleNames">

        <add value="addressbook"/>

    </putList>

    <putList name="moduleLinks">

        <add value="#" />

    </putList>

</definition>

```

module1 above is configured to read the struts definition, /personaladdressbook.do, defined in struts-config-addressbook.xml:

```

<!-- ===== Action Mapping Definitions ===== -->

    <action-mappings
type="com.edocs.application.tbm.core.common.EdocsActionMapping">

        <!--

***** -->

        <!-- ***** LDE ACTIONS ***** -->

        <!--

***** -->

        <action path="/personaladdressbook"
type="com.edocs.application.tbm.b2b.actions.PersonalAddressBookRequestAction"
name="addressBookForm" scope="request" validate="false">

```

```

-->
                                <!-- <set-property property="checkLogin" value="false" /> -
                                <forward name="success"
path=".module.addressbook.personaladdressbook"/>
                                </action>

```

The Struts action class, *com.edocs.application.tbm.b2b.actions.PersonalAddressBookRequestAction*, loads the address book view into another tile, *.module.addressbook.personaddressbook*, defined in *tiles-defs-addressbook.xml*:

```

<definition name=".module.addressbook.personaladdressbook"
extends=".view.setup">
    <put name="viewName" value="Pab_Display"/>
    <put name="viewType" value="HTML"/>
    <put name="ddn" value="PersonalAddressBook"/>
    <put name="useExistingViewInfo" value="true"/>
</definition>

```

This tile extends *.view.setup*, which is defined in *tiles-defs.xml*:

```

<!-- Views Setup (Base definition for rendering the views created by eadirect) --
>
    <definition name=".view.setup" path="/_templates/display_view.jsp"
controllerClass="com.edocs.application.tbm.displayview.DisplayTiledView">
        <put name="viewName" value="DefaultView"/>
        <put name="viewType" value="HTML"/>
        <put name="ddn" value="TBM"/>
    </definition>

```

Here are other popup tiles available in *tiles-defs-addressbook.xml*. Note that you can also define a list of module names and links for the session here.

```

<!-- ***** -->
    <!-- ***** PERSONAL ADDRESS BOOK popup tiles
***** -->
    <!--
***** -->
    <definition name=".main.addressbook.pab.add"
extends=".template.popup">
        <put name="module1" value=".module.addressbook.pab.add"/>
    </definition>
    <definition name=".module.addressbook.pab.add"
extends=".template.module">
        <put name="form" value="/addressbook/pab-add.jsp" type="page"/>
        <put name="activeModule" value="addressbook"/>

```

```

        <putList name="moduleNames">
            <add value="addressbook"/>
        </putList>
        <putList name="moduleLinks">
            <add value="#"/>
        </putList>
    </definition>
    <definition name=".main.addressbook.pab.edit"
extends=".template.popup">
        <put name="module1" value=".module.addressbook.pab.edit"/>
    </definition>
    <definition name=".module.addressbook.pab.edit"
extends=".template.module">
        <put name="form" value="/addressbook/pab-edit.jsp"
type="page"/>
        <put name="activeModule" value="addressbook"/>
        <putList name="moduleNames">
            <add value="addressbook"/>
        </putList>
        <putList name="moduleLinks">
            <add value="#"/>
        </putList>
    </definition>
    <definition name=".main.addressbook.pab.addnew"
extends=".template.popup">
        <put name="module1" value=".module.addressbook.pab.addnew"/>
    </definition>
    <definition name=".module.addressbook.pab.addnew"
extends=".template.module">
        <put name="form" value="/addressbook/pab-addnew.jsp"
type="page"/>
        <put name="activeModule" value="addressbook"/>
        <putList name="moduleNames">
            <add value="addressbook"/>
        </putList>
        <putList name="moduleLinks">
            <add value="#"/>
        </putList>
    </definition>

```

The jsp pages in the tile definitions above often point to two submit actions, “show.do” and “save.do”, as defined in struts-config-addressbook:

```

<!-- ***** -->

    <!-- ***** PERSONAL ADDRESS BOOK
***** -->

    <!--
***** -->

        <!-- Show the PAB Add Page -->

            <action path="/addressbook/pab-add/show"
type="com.edocs.application.tbm.b2b.actions.PersonalShowAddressAction"
name="addressBookForm" scope="request" parameter=".main.addressbook.pab.add"
validate="false">

                <!-- <set-property property="checkLogin" value="false"
/> -->

                    <forward name="success"
path=".main.addressbook.pab.add"/>

            </action>

            <!-- Save the new PAB -->

                <action path="/addressbook/pab-add/save"
type="com.edocs.application.tbm.b2b.actions.PersonalSaveAddressAction"
name="addressBookForm" scope="request" parameter="/addressbook/pab-
complete.jsp" validate="true" input="pab_add">

                    <!-- <set-property property="checkLogin" value="false"
/> -->

                        <forward name="success" path="/addressbook/pab-
complete.jsp"/>

                </action>

                <!-- Show the PAB Edit Page -->

                    <action path="/addressbook/pab-edit/show"
type="com.edocs.application.tbm.b2b.actions.PersonalShowAddressAction"
name="addressBookForm" scope="request" parameter=".main.addressbook.pab.edit"
validate="false">

                        <!-- <set-property property="checkLogin" value="false"
/> -->

                            <forward name="success"
path=".main.addressbook.pab.edit"/>

                    </action>

                    <!-- Save the PAB Edit -->

                        <action path="/addressbook/pab-edit/save"
type="com.edocs.application.tbm.b2b.actions.PersonalSaveAddressAction"
name="addressBookForm" scope="request" parameter="/addressbook/pab-
complete.jsp" validate="true" input="pab_edit">

                            <!-- <set-property property="checkLogin" value="false"/>
-->

                                <forward name="success" path="/addressbook/pab-
complete.jsp"/>

                        </action>

                        <!-- Show the PAB AddNew Page -->

                            <action path="/addressbook/pab-addnew/show"
type="com.edocs.application.tbm.b2b.actions.PersonalShowAddressAction"
name="addressBookForm" scope="request"
parameter=".main.addressbook.pab.addnew" validate="false">

```

```

                                <!-- <set-property property="checkLogin" value="false"
/> -->

                                <forward name="success"
path=".main.addressbook.pab.addnew"/>
                                </action>
                                <!-- Save the PAB Delete -->
                                <action path="/addressbook/pab-delete/save"
type="com.edocs.application.tbm.b2b.actions.PersonalDeleteAddressAction"
name="addressBookForm" scope="request" validate="false" input="pab">
                                <!-- <set-property property="checkLogin" value="false"
/> -->

                                <forward name="success"
path="/profile/personal_address_book.do"/>
                                </action>

```

Note that the form used in these jsp pages is configured in a form-bean in struts-config-addressbook.xml. Read the javadocs for `com.edocs.application.tbm.b2b.forms.AddressBookForm`:

```

<struts-config>
    <!-- ===== Form Bean Definitions
===== -->

    <form-beans>
        <form-bean name="addressBookForm"
type="com.edocs.application.tbm.b2b.forms.AddressBookForm"/>
    </form-beans>
...

```

To customize a personal address book for the TBM application, read the javadocs for the `tbm-b2b-app` application, paying particular attention to the Struts action and form classes delivered for personal address books. Open the WAR file delivered with the `tbm-b2b-app` ear to see the tiles delivered in `webroot/_templates` and `webroot/addressbook`. Also see the Struts and Tiles configuration files delivered in `webroot/WEB-INF`.

Extend or use `com.edocs.application.tbm.b2b.forms.AddressBookForm` and `com.edocs.application.tbm.b2b.actions.Personal*AddressActions` classes as needed. These classes employ the `AddressBook` component delivered with TBM. To customize the `AddressBook` component, read the javadocs delivered for the `com.edocs.domain.telco.addrbook` classes. See Section 11.3, `AddressBook` Component for a description of lower level services provided by TBM.

11.3 Corporate Address Books

Corporate address books follow the same application architecture used for personal address books as discussed in Section 11.1 above. See corresponding corporate address book sections defined in the configuration files described above.

11.4 AddressBook Component

The AddressBook component services both personal and corporate address book applications. The AddressBook component is delivered in the TBM EAR. Read the javadocs delivered for `com.edocs.domain.telco.addrbook` for implementation details.

The AddressBook component consists of the following classes:

`com.edocs.domain.telco.addrbook.ABException` (exception class for addressbook use)

`com.edocs.domain.telco.addrbook.AddrBookViewProcessor` (extends `RUIViewProcessor` and provides address tag substitution capability for views)

`com.edocs.domain.telco.addrbook.AddressBook` (abstract superclass of `PersonalAddressBook` and `CorporateAddressBook`)

`com.edocs.domain.telco.addrbook.AddressBookFactory` (gets/creates personal or corporate address books from/for the session)

`com.edocs.domain.telco.addrbook.AddressBookRequestZLDEStream` (creates an `LDEToolkit` stream given a `ddn`, account number, and addressbook)

`com.edocs.domain.telco.addrbook.AddressBookViewHelper` (provides stream filtering)

`com.edocs.domain.telco.addrbook.AddressEntry` (an address)

`com.edocs.domain.telco.addrbook.CABDataSource` (corporate address book data source)

`com.edocs.domain.telco.addrbook.CorporateAddressBook` (extends `AddressBook`)

`com.edocs.domain.telco.addrbook.PABDataSource` (personal address book data source)

`com.edocs.domain.telco.addrbook.PersonalAddressBook` (extends `AddressBook`)

The Struts action classes delivered in the TBM application WAR were described in Section 11.1 in the context of their interface with personal address book tiles. Here we describe their interface with the AddressBook component. The Struts action classes involved:

`com.edocs.application.tbm.b2b.actions.AddressActionBase`

`com.edocs.application.tbm.b2b.actions.PersonalAddressBookRequestAction`

`com.edocs.application.tbm.b2b.actions.PersonalShowAddressAction`

`com.edocs.application.tbm.b2b.actions.PersonalSaveAddressAction`

`com.edocs.application.tbm.b2b.actions.PersonalDeleteAddressAction`

`com.edocs.application.tbm.b2b.actions.CorporateAddressBookRequestAction`

`com.edocs.application.tbm.b2b.actions.CorporateShowAddressAction`

`com.edocs.application.tbm.b2b.actions.CorporateSaveAddressAction`

`com.edocs.application.tbm.b2b.actions.CorporateDeleteAddressAction`

These action classes extend the TBM core component, `EdocsAction`. These action classes call the `AddressBook` component to configure and load an addressbook into an e-Statement view (`PersonalAddressBookRequestAction`) or to show (`PersonalShowAddressAction`), save (`PersonalSaveAddressAction`), or delete (`PersonalDeleteAddressAction`) addresses from the address book.

11.4.1 Displaying the Address Book

The `PersonalAddressBookRequestAction` is configured in the action mapping `/personaladdressbook` in `struts-config-addressbook.xml`. This action class loads the personal address book into a view as follows:

```
try {
    log.debug("Entering PersonalAddressBookRequestAction.doAction()");
    LDEManager ldeManager = LDEManager.getInstance();
    AddressBook addressBook =
AddressBookFactory.getSessionPersonalAddressBook(httpServletRequest);
    ICustomer customer = CAMClassFactory.getCustomer(httpServletRequest);
    String accountNumber =
customer.getCurrentAccount().getAccountNumber();
    String ddn = customer.getDDNMap().getDDN("PersonalAddressBook");
    log.debug("addressbook ddn " + ddn);

    AddressBookRequestZLDEStream ldeStream = new
AddressBookRequestZLDEStream(ddn, accountNumber, addressBook);

    String docId = ldeManager.setContent(ldeStream);

    ViewInfo viewInfo = new ViewInfo();
    viewInfo.setDdn(ddn);
    viewInfo.setDocId(docId);
    viewInfo.setViewName("Pab_Display");
    viewInfo.setViewType("HTML");

    SessionUtils.setView(httpServletRequest, viewInfo);
}
```

This action instantiates an `LDEToolkit` component, `com.edocs.domain.telco.lde.LDEManager`, and the `AddressBook` component, `AddressBook`. It uses the CAM layer component, `CAMClassFactory`, to get the current customer, account number, and personal address book DDN. It instantiates a live data extraction (LDE) stream (`AddressBookRequestZLDEStream`) which it uses to set the view content. `ViewInfo` parameters are populated and the view is set in the session for display. (For more information about LDE, see Chapter 3.4 Live Data Extraction.)

Once the personal address book is loaded and ready for display, the action class forwards to the “success” jsp tile defined in struts-config-addressbook:

```
<!-- ***** -->

<!-- ***** LDE ACTIONS ***** -->

<!--
***** -->

<action path="/personaladdressbook"
type="com.edocs.application.tbm.b2b.actions.PersonalAddressBookRequestAction"
name="addressBookForm" scope="request" validate="false">

    <!-- <set-property property="checkLogin" value="false"
/> -->

    <forward name="success"
path=".module.addressbook.personaladdressbook"/>
</action>
```

.module.addressbook.personaladdressbook in tiles-defs-addressbook extends .view.setup from tiles-defs.xml. Finally, webroot/_templates/display_view.jsp is called to display the personal address book view.

(From tiles-defs.xml)

```
<!-- Views Setup (Base definition for rendering the views created by eStatement)
-->

<definition name=".view.setup" path="/_templates/display_view.jsp"
controllerClass="com.edocs.application.tbm.displayview.DisplayTiledView">

    <put name="viewName" value="DefaultView"/>
    <put name="viewType" value="HTML"/>
    <put name="ddn" value="TBM"/>

</definition>
```

(From tiles-defs-addressbook.xml)

```
<definition name=".module.addressbook.personaladdressbook" extends=".view.setup">

    <put name="viewName" value="Pab_Display"/>
    <put name="viewType" value="HTML"/>
    <put name="ddn" value="PersonalAddressBook"/>
    <put name="useExistingViewInfo" value="true"/>

</definition>
```

11.4.2 Showing Addresses

When an address is selected from the personal address book view, the Struts action class, `PersonalShowAddressAction`, instantiates `AddressBookForm` and calls the `AddressBookComponent`, `AddressBookFactory`, to get the current personal address book from the session or create one, if it does not exist. `PersonalAddressBook` gets an instance of `PABDataSource` to persist the address book. Then it passes an `AddressBookForm` and an `AddressBook` as parameters in a call to `AddressActionBase.populateForm()`, which calls `PersonalAddressBook.getEntry()` to instantiate another `AddressBook` component, `AddressEntry`. Once the form is populated, the Struts action forwards to `addressbook/pab-add.jsp`.

See the javadocs delivered for the `AddressBook` component for more implementation details.

11.4.3 Saving and Deleting Addresses

After an address has been shown, it may be updated, saved, or deleted. A submit button on the `addressbook/pab-add.jsp` page forwards to a `save.do` Struts action.

The Struts actions, `PersonalDeleteAddressAction` and `PersonalSaveAddressAction`, instantiate `AddressBookForm` and call the `AddressBookFactory` to get the current personal address book from the session. Then the action classes pass these parameters in calls to `AddressActionBase.updateAddressBook()`, which deletes, updates, or saves new `AddressEntry` information to the `PABDataSource` via `PersonalAddressBook.deleteEntry()`, `PersonalAddressBook.updateEntry()`, and `PersonalAddressBook.addEntry()`.

See the javadocs delivered for the `AddressBook` component for more implementation details.

11.4.4 Address Book View Processing

The `AddressBook` component provides an address book processor for filtering address book views. LDE view streams are strings that may be manipulated to change presentation in certain ways. The `AddressBook` component, `AddrBookViewProcessor`, takes a `ViewInfo` object, processes the input stream from the `ViewInfo` object inserting address book entries where address book tags appear (via `AddrBookViewHelper`), and then returns a new `ViewInfo` object with the address book entries inserted.

`AddressBookViewHelper` creates a “(Don’t) Show Nicknames” toggle link on the address book view and substitutes phone numbers with nickname hyperlinks, if the nickname function is toggled on. See `webroot/_assets/scripts/addrbook.js` for the javascript functions.

11.4.5 Address Book Post-Processing of TBM Views

As for address book views, support for substitution of phone numbers with nickname hyperlinks is provided for all TBM views. See the javadocs delivered for the `DisplayView` component and Chapter 3, *Statementing and Content Access* for details.

To configure a view to use an address book post processor, set name="postprocessor" and name="usePersonalAddressBook" in tiles-defs.xml as follows:

```
<!--View Display Detailed Usage (Home) Collapsed-->
    <definition name=".statement.detailedUsage.view"
extends=".view.setup">
        <put name="viewName" value="B2B_DetHomeCol"/>
        <put name="usePersonalAddressBook" value="true"/>
        <put name="postProcessor" value="AddressBook"/>
        <put name="useViewInfoForFilters" value="true"/>
        <put name="subIdx" value="true"/>
    </definition>
```

The AddressBookPostProcessor resides in the DisplayView component package, com.edocs.application.tbm.displayview. Override this class as needed and configure it in app-config.properties:

```
#post processor classes

PostProcessors=PostProcessEngine
PostProc.PostProcessEngine=com.~.displayview.PostProcessorComposition
PostProcessor.classes=addressbook.class
addressbook.class=com.~.displayview.AddressBookPostProcessor
```


This chapter describes how to create custom job types that include the Shell Command Task. This task allows you to run an external command script to process the output files from other tasks within the job.

You can use this chapter to:

- Define a custom job type for the Command Center and create a SQL script, containing job type and task information, to add the new job type.
- View and configure the new job type in the Command Center.

12.1 About Jobs and the Shell Command Task

TBM has several predefined job types available in its Command Center. Each job is made up of one or more tasks. For complete listing of jobs and tasks, see the *TBM Administration Guide*.

However, there may be times when you will want to expand these predefined Jobs to fit your needs. For cases like this TBM has the ability to define your own **custom Job** Type that you can make up from a combination of the predefined tasks that come with TBM and/or your own custom task by defining what is referred to in TBM as a **Shell Command Task**.

A Shell Command Task is a way of invoking a shell script, executable, or other program that was written to perform a task specific to your requirements. It enables you to run custom scripts or programs, such as pre- or post- processors as part of a user-defined job. You can create your own Job Type by creating a SQL script that updates the database. Once the database is updated this Job Type becomes available to you via the Command Center. The new Job Type can then be configured, scheduled, and run from within the Command Center.

For example, you could create a new custom job called **Preprocess** to run a pre-processor on the input file in an Indexer job. At runtime, the **Preprocess** task would be inserted between the Scanner and the Indexer tasks. Another use would be to create a job to run a validation engine (sum all amount due, for example) on the output of the Indexer task. At runtime, the **SumAllDue** task would be inserted between the **Indexer** and the **IXLoader** tasks.

The following illustration shows a new custom Indexer job type in the Command Center for the latter example.

Task 2: Indexer	
DDF Path:	/opt/EDCSbd/AppProfiles/Alex2/DOC_CONFIG/Indexer/20010319155326/
Doc Date:	-----Today's Date-----
Index Field List:	Name* EndDate* PaymentDueDate* >> << TotalCharges* StartDate* TaxesAndSurcharges* AmountReceived* StatementDate*
Task 3: ShellCmdTask	
Shell Command:	enter your shell command here
Environment Vars:	enter semi-colon separated vars here
Task 4: IXLoader	
Skip Rows:	0
Split Size:	0
Optional Field Count:	0
Load Method:	Direct
Task 5: AutoIndexVolAccept	
Action on Index Volume:	Auto Accept

12.2 Defining a New Job Type

This section includes information about:

- Creating the job type script
- Configuring the new job type
- Examples of the job type script

12.2.1 Create the Job Type Script

To create a job type you create a single SQL script to run in the TBM database using the Oracle utility **sqlplus**. Within this SQL script you define:

1. The job name
2. The tasks and the order in which they will run
3. The input arguments

The following sections provide a detailed topic description of each part. Each section uses the example of specifying a new job type that is similar to the current Indexer job, except that between scanning for an input file (Scanner Task) and actually indexing the file (Indexer Task) you need to invoke a preprocessor to modify the input file. This is the situation where you need to insert the **ShellCmdTask** between the other tasks.

Example *sqlplus* script for Oracle

```
DECLARE jtid NUMBER;
BEGIN

    -- Define the job name
    jtid := pwc_job_types.create_job_type ('myIndexer');

    -- Specify the job tasks and their order
    pwc_job_types.create_job_type_task(jtid,'Scanner', 1);
    pwc_job_types.create_job_type_task(jtid,'ShellCmdTask', 2);
    pwc_job_types.create_job_type_task(jtid,'Indexer', 3);
    pwc_job_types.create_job_type_task(jtid,'IXLoader', 4);
    pwc_job_types.create_job_type_task(jtid, 'AutoIndexVolAccept', 5);

    -- Define the tasks input arguments
    pwc_job_types.create_job_type_io(jtid,'ShellCmdTask', 'input params', 'INPUT',
    2,'Scanner', 'output file name', 'OUTPUT', 1);

    pwc_job_types.create_job_type_io(jtid, 'Indexer','data file name', 'INPUT', 3,
    'ShellCmdTask','shell output', 'OUTPUT', 2);

    pwc_job_types.create_job_type_io(jtid, 'Indexer', 'ddn volume number', 'INPUT',
    3,'Scanner', 'ddn volume number', 'OUTPUT', 1);

    pwc_job_types.create_job_type_io(jtid, 'IXLoader', 'index volume number',
    'INPUT', 4,'Scanner', 'ddn volume number', 'OUTPUT', 1);

    pwc_job_types.create_job_type_io (jtid, 'IXLoader', 'ir file name', 'INPUT',
    4, 'Indexer', 'ir file name', 'OUTPUT', 3);

    pwc_job_types.create_job_type_io(jtid, 'AutoIndexVolAccept', 'index volume
    number', 'INPUT', 5, 'Scanner', 'ddn volume number', 'OUTPUT', 1);

END;
```

Example script for AIX/DB2

To create a DB2 shell command for a custom job in AIX, run the following command:

```
db2 -td@ -vf customjob.sh
```

where **customjob.sh** is the name of a shell script customized for your job, platform, and environment. See the example below for a sample script to customize.

```
DROP PROCEDURE db2inst1.tmp_pwc_jtt_sp() @
CREATE PROCEDURE db2inst1.tmp_pwc_jtt_sp()
LANGUAGE SQL
BEGIN
    DECLARE jtid            INTEGER;
    DECLARE l_job_type_name VARCHAR(32);
    DECLARE l_task_name     VARCHAR(32);
    DECLARE l_task_order    INTEGER;
```

```

DECLARE l_i_task_name      VARCHAR(32);
DECLARE l_i_task_io_name  VARCHAR(32);
DECLARE l_i_task_io_type  VARCHAR(32);
DECLARE l_i_task_order    INTEGER;
DECLARE l_o_task_name      VARCHAR(32);
DECLARE l_o_task_io_name  VARCHAR(32);
DECLARE l_o_task_io_type  VARCHAR(32);
DECLARE l_o_task_order    INTEGER;

-- job type with
'Scanner': 'ShellCmdTask': 'Indexer': 'IXLoader': 'AutoIndexVolAccept'

SET l_job_type_name = 'Custom_Indexer';
CALL pwc_job_types.create_job_type(jtid, l_job_type_name);

SET l_task_name = 'Scanner';
SET l_task_order = 1;
CALL pwc_job_types.create_job_type_task(jtid, l_task_name, l_task_order);
SET l_task_name = 'ShellCmdTask';
SET l_task_order = 2;
CALL pwc_job_types.create_job_type_task(jtid, l_task_name, l_task_order);
SET l_task_name = 'Indexer';
SET l_task_order = 3;
CALL pwc_job_types.create_job_type_task(jtid, l_task_name, l_task_order);
SET l_task_name = 'IXLoader';
SET l_task_order = 4;
CALL pwc_job_types.create_job_type_task(jtid, l_task_name, l_task_order);
SET l_task_name = 'AutoIndexVolAccept';
SET l_task_order = 5;
CALL pwc_job_types.create_job_type_task(jtid, l_task_name, l_task_order);

SET l_i_task_name      = 'ShellCmdTask';
SET l_i_task_io_name   = 'input params';
SET l_i_task_io_type   = 'INPUT';
SET l_i_task_order     = 2;
SET l_o_task_name      = 'Scanner';
SET l_o_task_io_name   = 'output file name';
SET l_o_task_io_type   = 'OUTPUT';
SET l_o_task_order     = 1;
CALL pwc_job_types.create_job_type_io(jtid, l_i_task_name, l_i_task_io_name,
l_i_task_io_type, l_i_task_order, l_o_task_name, l_o_task_io_name,
l_o_task_order);
SET l_i_task_name      = 'Indexer';

```

```

SET l_i_task_io_name = 'data file name';
SET l_i_task_io_type = 'INPUT';
SET l_i_task_order   = 3;
SET l_o_task_name     = 'ShellCmdTask';
SET l_o_task_io_name  = 'shell output';
SET l_o_task_io_type  = 'OUTPUT';
SET l_o_task_order    = 2;

CALL pwc_job_types.create_job_type_io(jtid, l_i_task_name, l_i_task_io_name,
l_i_task_io_type, l_i_task_order, l_o_task_name, l_o_task_io_name,
l_o_task_io_type,
l_o_task_order);

SET l_i_task_name     = 'Indexer';
SET l_i_task_io_name  = 'ddn volume number';
SET l_i_task_io_type  = 'INPUT';
SET l_i_task_order    = 3;
SET l_o_task_name     = 'Scanner';
SET l_o_task_io_name  = 'ddn volume number';
SET l_o_task_io_type  = 'OUTPUT';
SET l_o_task_order    = 1;

CALL pwc_job_types.create_job_type_io(jtid, l_i_task_name, l_i_task_io_name,
l_i_task_io_type, l_i_task_order, l_o_task_name, l_o_task_io_name,
l_o_task_io_type,
l_o_task_order);

SET l_i_task_name     = 'IXLoader';
SET l_i_task_io_name  = 'index volume number';
SET l_i_task_io_type  = 'INPUT';
SET l_i_task_order    = 4;
SET l_o_task_name     = 'Scanner';
SET l_o_task_io_name  = 'ddn volume number';
SET l_o_task_io_type  = 'OUTPUT';
SET l_o_task_order    = 1;

CALL pwc_job_types.create_job_type_io(jtid, l_i_task_name, l_i_task_io_name,
l_i_task_io_type, l_i_task_order, l_o_task_name, l_o_task_io_name,
l_o_task_io_type,
l_o_task_order);

SET l_i_task_name     = 'IXLoader';
SET l_i_task_io_name  = 'ir file name';
SET l_i_task_io_type  = 'INPUT';
SET l_i_task_order    = 4;
SET l_o_task_name     = 'Indexer';
SET l_o_task_io_name  = 'ir file name';
SET l_o_task_io_type  = 'OUTPUT';
SET l_o_task_order    = 3;

CALL pwc_job_types.create_job_type_io(jtid, l_i_task_name, l_i_task_io_name,

```

```

l_i_task_io_type, l_i_task_order, l_o_task_name, l_o_task_io_name,
l_o_task_io_type,
l_o_task_order);

SET l_i_task_name      = 'AutoIndexVolAccept';
SET l_i_task_io_name = 'index volume number';
SET l_i_task_io_type = 'INPUT';
SET l_i_task_order    = 5;
SET l_o_task_name      = 'Scanner';
SET l_o_task_io_name = 'ddn volume number';
SET l_o_task_io_type = 'OUTPUT';
SET l_o_task_order    = 1;

CALL pwc_job_types.create_job_type_io(jtid, l_i_task_name, l_i_task_io_name,
l_i_task_io_type, l_i_task_order, l_o_task_name, l_o_task_io_name,
l_o_task_io_type,
l_o_task_order);

END @

CALL db2inst1.tmp_pwc_jtt_sp() @
DROP PROCEDURE db2inst1.tmp_pwc_jtt_sp() @

```

Name the Job

The first part of the script is to give your new task a name. The syntax to do this is:

```
jtid := pwc_job_types.create_job_type ('<new_job_name>');
```

In the script, the **create_job_type** call defines a unique job type ID (**jtid**) for the new *Indexer1* job type.

So if your new job name is **myIndexer**, then the code script will be:

```
jtid := pwc_job_types.create_job_type ('myIndexer');
```

Specify Job Tasks

The next step is to specify what tasks will be part of the new job, and in what order will they execute. The syntax is:

```
pwc_job_types.create_job_type_task(jtid, '<task_name>', n);
```

where *n* equals the order number of the task and *jtid* is the job type id – created with **pwc_job_types.create_job_type()** function. The **create_job_type_task** call defines the order of the tasks in the job.

In the above example, the plan is to create a new job type based on the current Indexer job type. The tasks included in the Indexer Job are (in their order of execution):

- Scanner
- Indexer
- IXLoader
- AutoIndexVolAccept

If you insert the **ShellCmdTask** after the Scanner Task, it will become task 2, and the others will be incremented by one. The code example is:

```
pwc_job_types.create_job_type_task(jtid, 'Scanner', 1);
pwc_job_types.create_job_type_task(jtid, 'ShellCmdTask', 2);
pwc_job_types.create_job_type_task(jtid, 'Indexer', 3);
pwc_job_types.create_job_type_task(jtid, 'IXLoader', 4);
pwc_job_types.create_job_type_task(jtid, 'AutoIndexVolAccept', 5);
```

Define Input Arguments

Each task has input and output arguments, and a particular task may require the output arguments from a previous task to function properly. For example, in the default Indexer job, its Indexer task takes two input arguments from the Scanner Task. In the SQL Script you define which specific input arguments for a task are used from the specific output arguments from another task.

For a list of arguments, see the Javadoc for the *SDK API Specification*.

To define the input and output parameters, the following is the syntax of the function of the call that uses nine arguments:

```
pwc_job_types.create_job_type_io(jtid,
                                '<input_task_name>',
                                '<input_argument>',
                                'INPUT',
                                x,
                                '<output_task_name>',
                                '<output_argument>',
                                'OUTPUT',
                                y);
```

where *x* is the order number of the input task and *y* is the order number of the output task. The **create_job_type_io** calls define the input values for each job task. It accepts the following parameter values:

- The job type ID (**jtId**)
- The task name receiving the input value
- The input parameter name
- The I/O type (**INPUT**)
- The order number for the task receiving the input value (defined earlier in the script)
- The previous task name dispensing the output to be used for input
- The output parameter name from the previous task
- The I/O type (**OUTPUT**)
- The order number of the task dispensing the output value (defined earlier in the script)

The following breaks down the input arguments used in the above example script:

```
pwc_job_types.create_job_type_io(jtid,
                                'ShellCmdTask',
                                'input params',
                                'INPUT',
                                2,
                                'Scanner',
                                'output file name',
                                'OUTPUT',
                                1);
```

The input argument *input params* for the **ShellCmdTask** uses the output argument *output file name* from the Scanner task.

```
pwc_job_types.create_job_type_io(jtid,
                                'Indexer',
                                'data file name',
                                'INPUT',
                                3,
                                'ShellCmdTask',
                                'shell output',
                                'OUTPUT',
                                2);

pwc_job_types.create_job_type_io(jtid,
                                'Indexer',
                                'ddn volume number',
                                'INPUT',
                                3,
                                'Scanner',
                                'ddn volume number',
                                'OUTPUT',
                                1);
```

The input arguments *data file name* and *ddn volume number* for the Indexer task uses the output arguments *shell output* from the **ShellCmdTask** and *ddn volume number* from the Scanner task respectively.

```
pwc_job_types.create_job_type_io(jtid,
                                'IXLoader',
                                'index volume number',
                                'INPUT',
                                4,
                                'Scanner',
                                'ddn volume number',
                                'OUTPUT',
                                1);

pwc_job_types.create_job_type_io(jtid,
                                'IXLoader',
                                'ir file name',
                                'INPUT',
                                4,
                                'Indexer',
                                'ir file name',
                                'OUTPUT',
                                3);
```

The input arguments *index volume number* and *ir file name* for the **IXLoader** task uses the output arguments *ddn volume number* from the Scanner and *ir file name* from the Indexer respectively.

```
pwc_job_types.create_job_type_io(jtid,
                                'AutoIndexVolAccept',
                                'index volume number',
                                'INPUT',
                                5,
                                'Scanner',
                                'ddn volume number',
                                'OUTPUT',
                                1);
```

The input argument *index volume number* for the **AutoIndexVolAccept** task uses the output argument *ddn volume number* from the Scanner task.

12.2.2 Configuring Your New Job Type

After creating the script, you need to run it against the Oracle database used by TBM (as described in the Installation and Configuration Guides). For example, if the script is named **myindexer.sql** and placed in **/opt/EDCSbd/db** (the default database location for TBM), you could run the following in SQL*Plus:

```
$ sqlplus -s edx_dba/edx@edx.db @ /opt/EDCSbd/db/myindexer.sql
```

The above command presumes you are using the default names for the TBM database (**edx0**) and database administrator/password (**edx_dba/edx**).



Tip

Before the new job type is available in the Command Center, you have to stop and start Your application server after running the script.

Once the new job type is available to you in the Command Center, you can define the new job using that new job type.

Define the Shell Command Task

If you have included the **ShellCmdTask** with your new job type, it has 2 input fields to define:

- Shell Command
- Environment variables

The Shell Command field defines the location of the shell script to execute on your system. Note that the user that starts the application server must have read/execute permissions for that location.

The shell command must output, on its standard output, the name of its output file that is the input file to be processed by the next task in the job. If the shell command doesn't output any file name, the job stops as a no-op. If it is successful, the shell command must set its exit code to 0.

If the shell command fails, it must set its exit code to a non-zero value. Additionally, it may output, on its standard error, a message describing the failure. The error message will be logged into the log file by TBM. However, any errors within the shell command are not logged and must be handled separately.

For example, the following shell command would be useful after the Scanner task to ensure Windows files have the correct format for UNIX:

```
#!/bin/csh
# Preprocessor to run dos2unix on the input file

dos2unix $SHELL_INPUT $SHELL_INPUT.ux >& /dev/null
if ($status != 0) exit $status          # failure
echo $SHELL_INPUT.ux                   # new input file
exit 0                                  # success
```

The Environment variables field specifies the environment variables for the shell command. By default, the external command is passed the following environment variables:

- DDN - the name of the application to which the job belongs
- JOB_NAME - the name of the job to which the task is a part of.
- STATUS - the status of the job (has it been started, did it succeed/fail, etc).
- PREVIOUS_STATUS
- SHELL_INPUT - any input from a previous task. The SHELL_INPUT variable is only set if the shell command task is linked with another task in the context of a job. Otherwise it is null.

If your shell command requires any other environment variables, you'll need to specify them in this field.

12.2.3 Another Example of Defining a New Job Type

The following is another example that defines an index job called Indexer2 with the following tasks:

- Scanner
- Indexer
- ShellCmdTask
- IXLoader
- AutoIndexVolAccept

As mentioned in the previous section, a reason for this new job type could be to run a validation engine (sum all amount due for example) on the output of the Indexer task. If the amount due exceeds a certain amount, it may require a careful verification of the input data stream as described in the SDK Module: Auditing Data Streams with the Verify API.

For this case you can create the following SQL script:

```
DECLARE jtid NUMBER;
BEGIN

  jtid := pwc_job_types.create_job_type('Indexer2');
  pwc_job_types.create_job_type_task(jtid, 'Scanner', 1);
  pwc_job_types.create_job_type_task(jtid, 'Indexer', 2);
  pwc_job_types.create_job_type_task(jtid, 'ShellCmdTask', 3);
```

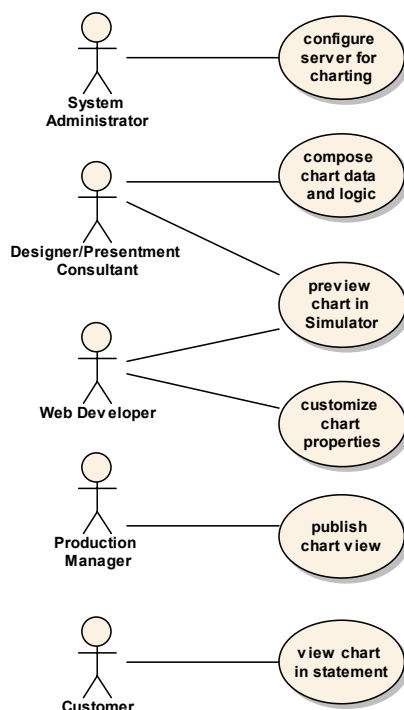
```
pwc_job_types.create_job_type_task(jtid, 'IXLoader', 4);
pwc_job_types.create_job_type_task(jtid, 'AutoIndexVolAccept', 5);
pwc_job_types.create_job_type_io(jtid, 'Indexer', 'data file name', 'INPUT', 2,
'Scanner', 'output file name', 'OUTPUT', 1);
pwc_job_types.create_job_type_io(jtid, 'Indexer', 'ddn volume number', 'INPUT',
2, 'Scanner', 'ddn volume number', 'OUTPUT', 1);
pwc_job_types.create_job_type_io(jtid, 'ShellCmdTask', 'input params', 'INPUT',
3, 'Indexer', 'ir file name', 'OUTPUT', 2);
pwc_job_types.create_job_type_io(jtid, 'IXLoader', 'index volume number',
'INPUT', 4, 'Scanner', 'ddn volume number', 'OUTPUT', 1);
pwc_job_types.create_job_type_io(jtid, 'IXLoader', 'ir file name', 'INPUT', 4,
'Indexer', 'ir file name', 'OUTPUT', 2);
pwc_job_types.create_job_type_io(jtid, 'AutoIndexVolAccept', 'index volume
number', 'INPUT', 5, 'Scanner', 'ddn volume number', 'OUTPUT', 1);
END;
```


13.1 Introduction and Components

TBM can format statement data as a graphical chart in a dynamic HTML page. Charts consist of the **chart data**, which must be a table or group with at least two data rows, and the **chart properties**, which specify the type, design, and layout of the chart graphic.

Charting can involve most of the actors in a typical TBM workflow. The following overview diagram highlights the main tasks in the charting process.

13.1.1 Charting Use Case Overview Diagram



To present charts in online statements:

1. The **system administrator** follows the steps in [Configuring Charting for Your Server](#) to set up the display device, permissions, and awareness on the application server rendering the charts, and to install any specified fonts.
2. The TBM application **designer** follows the steps in [Composing Charts in Statements](#) to insert a chart placeholder in the **Application Logic File (ALF)** with the Composer.
3. The web application **developer** or the **designer** follows the steps in [Customizing Chart Properties](#) to fine-tune the design, layout, and data presentation of the chart in the ALF and the **chart properties file**. Advanced designers and developers may extend the available chart properties for [Designing Custom Charts with the Charting Servlet](#).
4. During the design process, the **designer** or **developer** can follow the steps in [Simulating Charts](#) to preview the chart in a simulated online statement with the Simulator API.
5. When the chart data and design are finalized, the **production manager** follows the steps in [Publishing Charts](#) to include the chart(s) in any dynamic online statements processed through the TBM Command Center.
6. Once the chart view and any associated views are published, the **customer** can view the chart as part of an online statement, so that charts will refresh dynamically with each new version set of statement data.

The sections in this chapter describe each of these tasks in detail.

13.1.2 Components of Charting

- Indexed data source (DDN and Indexer job)
- Application Logic File (ALF)
- Chart Properties File (*.properties)
- Simulator API
- HTML Web View for a Charting ALF
- Chart View for each Chart

13.2 Configuring Charting for Your Server

13.2.1 About Servers and Charting

The server rendering the charts, not the machine viewing the statement, determines font sizes and styles in charts. The server displaying charts must also have access **permissions** set to display charts, and requires **awareness** of an actual or virtual **display** device. This chapter discusses:

- Fonts
- Configuring a Headless Server for Charting
- Display Devices and xvfb
- Display Permissions and xhost
- Display Awareness



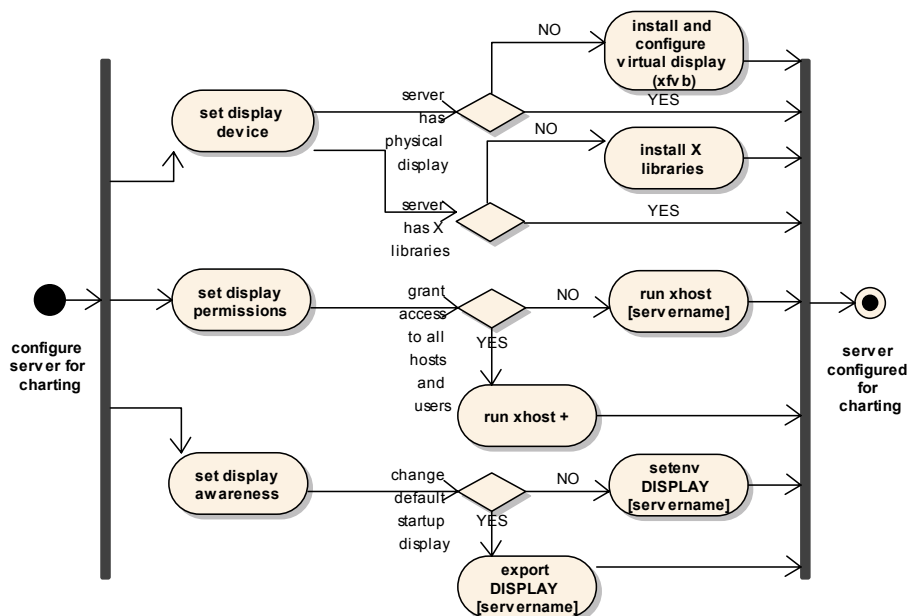
The configuration steps in this chapter apply primarily to **deployment** servers. Servers in a **production** environment often have physical display devices with graphics support, so that configuration may not be an issue. **Always test charts** (with the rest of your web application) **on your deployment platform**, and make any needed configurations for your charts to display properly with the correct fonts and styles.

13.2.2 About Fonts

Charts require graphics utilities and fonts that vary across platforms. **Windows NT/2000** has rich support for both graphics and fonts. UNIX systems like **Solaris** and **AIX** support graphics with an **X server**, or by using a virtual display, for example **xvfb**. Either option can offer rich font and style support, depending on fonts installed.

Any fonts you reference in your chart properties must be available *on your deployment server*, not on the machine where your browser views the charts. If you receive “font not found” or similar error messages when charting, check the fonts and styles available on your X server against those in your chart properties file.

13.2.3 Configuration Activity Diagram



13.2.4 Setting Display Devices and xvfb

Like other Java graphics packages, edocs charting extends the [java.awt](#) interface, which contains all of the classes for creating user interfaces and for painting graphics and images. These classes in turn require X libraries and access to an X display. To display charts properly, the web server rendering the charts must have a real or virtual X display device and the necessary X libraries.

In a development environment, the web server may have a real physical display device attached and have X Libraries loaded. However, at a typical server host site, few if any of the racks of server machines are connected to a display, and system administrators may hesitate to load X libraries if they are not installed.

If your deployment environment does not have a physical display **and** X libraries, your “headless” server will need a virtual X display like **xvfb**. The X Virtual Frame Buffer (**xvfb**) is an X server that can run on machines without display hardware or input devices. It emulates a dumb framebuffer using virtual memory.

xvfb may already be installed on your Unix system, in `/usr/X11R6`. If not, you will need to obtain and install a copy.



Tip

Documentation for **xvfb** (`man xvfb`) is hard to find. Many versions of UNIX have no manual entry for **xvfb** or have it in the wrong place. The University of Texas has posted `man xvfb` version 1 at <http://dell5.ma.utexas.edu/cgi-bin/man-cgi?xvfb+1>. NOAA also has an excellent README.**xvfb** and a binary of **xvfb** at ftp://ferret.wrc.noaa.gov/special_request/xvfb/solaris

13.2.5 Setting Display Permissions and xhost

You can control access to your X server with the UNIX program **xhost**. This access control program can add and delete host names or user names to the list permitted to connect to the X server.

Tip

The privacy and security controls in **xhost** are generally sufficient for a single-user workstation environment. You may prefer to use a custom authentication system for stronger access control.

xhost is located in different places on different systems. Look in `/usr/openwin/bin` or `/usr/local/share` to start. edocs recommends that you add **xhost** to your environment PATH variable.

To grant X server display access to all available hosts and users, type:

```
xhost +
```

For other options, see [Syntax and Parameters](#).

xhost Syntax and Parameters

Security requires that **xhost** be run only from the controlling host. For workstations, this is the server machine. For X terminals, it is the login host. The command syntax is:

```
xhost [[+-]name ...]
```

Parameter	Description
[+]name	Adds the given host name or user name to the list allowed to connect to the X server. The plus sign is optional.
-name	Removes the given host name or user name from the list allowed to connect to the X server. Existing connections are not broken, but new connection attempts will be denied.
+	Turns off access control; grants access to all host names and user names, even if not on the X server list.
-	Turns on access control; restricts access to only those host names and user names on the X server list.
nothing	Typing xhost without arguments prints a message indicating whether access control is enabled and listing those allowed to connect. This is the only option available to machines other than the controlling host.

Caution

Use care in removing hosts and users. **xhost** allows you to remove the current machine, but then will not permit further connections, including attempts to add it back. You must then reset the server in order to allow local connections again.

13.2.6 Setting Display Awareness

When you use X Windows tools, you must assign the environment variable `DISPLAY` to point to your local workstation, or wherever you would like the windows from the X Windows application displayed. When you run an application or web server from the command line, your server will use the current `DISPLAY` environment variable.



Tip

If you are running an X server on a remote machine, and displaying the windows on your local machine, you may also have to run **xhost** on your local machine to allow windows to be opened there: **xhost +remote_machine**

UNIX users can change where windows are displayed with the shell commands **setenv DISPLAY** or **export DISPLAY**.

To change the default display awareness and permissions:

1. Advanced users can modify the startup script for your application server.

For WebLogic, the startup script is located at:

```
<WL_HOME>/config/mydomain/startWebLogic.sh
```

For WebSphere, the startup script is located at:

```
<WS_HOME>/bin/startupServer.sh
```

2. Insert the following lines in your startup script, where *MyServer:2.0* is the name of your display:

```
DISPLAY=MyServer:2.0
export DISPLAY
/usr/openwin/bin/xhost + webservername
```

3. Specifying the web server name limits the X DISPLAY 2.0 to connections from the specified server. If the web server name is omitted (**xhost +**), then any host machine can connect to X on the server.

For more information on working with application server scripts, see the *TBM Installation Guide*.

13.2.7 Configuring a Headless Server for Charting

If your deployment environment does not have a physical display **and** X libraries, your “headless” server will need a virtual X display like **xvfb**. For more information on display [devices](#), [permissions](#), and [awareness](#), see the previous sections.



for AIX

The X Windows client for AIX systems requires the X11 package, which comes with the O/S but is not installed by default. To check whether X11 is installed, run **smi t** and check the installed packages option for AIX Windows X11 libraries, or look in the default directory `/usr/lpp/X11`.

To enable charting on a “headless” server (Solaris):

Download **xvfb** from

`ftp://www.ferret.noaa.gov/special_request/xvfb/solaris/`

Install to **/usr/X11R6**. **xvfb** will be installed in the **/bin** directory.

Enable X display permission on your web server with the command **xhost +.**

To set the current display to use the frame buffer for graphics display, set your **DISPLAY** variable, for example:

```
DISPLAY=ella:1; export DISPLAY
```

This will send any graphics output going to display 1 to shared memory.

Run **xvfb** as a background process.

```
/usr/X11R6/bin/xvfb :1 -screen 0 800x600x24 &
```

The "&" will kill the command window and leave the task running in the background.

This procedure will create a virtual display at **:1.0** with a size of 800x600 pixels and a color depth of 24 bits. To ensure that your Java environment will draw to this display, you must set the **DISPLAY** environment variable to **:1.0** **before** invoking Java. If you receive an environment exception, try changing the color depth or screen size.



xvfb must be installed in the directory **/usr/X11R6**, as it looks in this directory for needed fonts. If these fonts are not found under **/usr/X11R6**, **xvfb** will fail.

13.3 Composing Charts in Statements

13.3.1 About Charting in the Composer

Web designers and developers can use the TBM **Composer** tool to define data objects and custom tags in HTML templates for TBM **applications**. Defining a **chart tag** for a **table** or **group** will display that data object as a graphical chart in the online statement.

The Composer GUI allows you to define only a few basic chart properties: a **chart type** of Pie, the X and Y-axes for data, and the width and height of the chart. Once you have created this “placeholder chart” in the Composer, you will customize the look and feel of the chart by [Customizing Chart Properties](#) in the **ALF** or the **chart properties file**.

For more information on working with the Composer, see the *TBM Administration Guide*.

13.3.2 Inserting a Chart Tag in the Composer

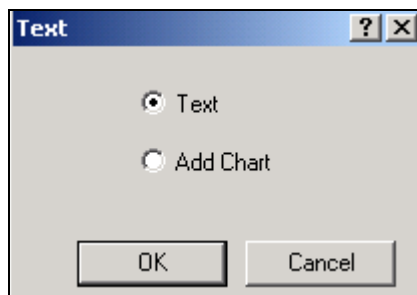
To chart data for any table in the DDF, you can drag and drop tables into the HTML template using the WYSIWYG or the HTML editor. You can represent a table as either a text table or as a chart.

**Tip**

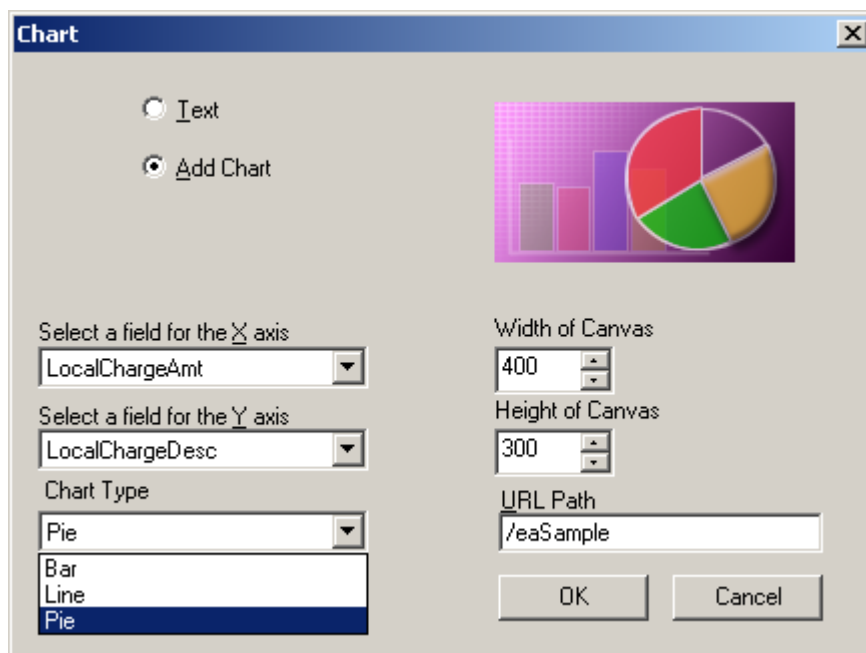
Tables to be charted must have **at least two fields**, one of which must contain **numeric values**. The Composer converts any values in non-numeric field types to numbers.

To insert a chart tag in the Composer:

1. Open an ALF in the Composer. For this example, open **NatlWireless\NW_LocSummary.alf**.
2. Click the **WYSIWYG** tab or **HTML** tab.
3. Click the Definition tab in the Tree.
4. Click to open Tables in the Tree.
5. Drag and drop the table  definitions into the HTML template. The table assumes the properties of the area in which it is placed. (Drag and drop the **LocalChargeSummary** table to the HTML template.)



6. Select **Chart**.



7. Select a field for the X-axis of the chart. (Select **LocalChargeAmount**.)
8. Select a field for the Y-axis of the chart. (Select **LocalChargeDesc**.)
9. Select the type of chart. (Select **Pie**.)

**Caution**

Pie charts are the only chart type available through the Composer. Selecting Bar or Line will still generate a Chart Type of Pie in the ALF and the chart properties file. For how to create chart types other than pie charts, see [Customizing Chart Properties](#).

10. Select the width and height settings for the chart. (Leave at 400 and 300 respectively.)
11. Enter the URL path to your web application root.
12. Click **OK**. (The tag **[E]LocalChargeSummary_0,U[/E]** appears.) This chart tag adds a placeholder for the chart to the HTML template.

**Tip**

Make a note of the name of the table you are charting, which appears in the chart tag. When you publish a chart view, you **must** name the view with this table name, in this example **LocalChargeSummary_0**. This name will also match the name of the chart properties file created by the Composer.

13. Delete any temporary placeholders in the HTML template, for example “XX.”
14. Click the **Save Template** icon .
15. Save the ALF by clicking the **Save ALF** icon .

Tip

When mapping a table to a template in the Composer, it is not necessary to encapsulate the table with HTML table row `<TR>` and table data `<TD>` (cell) tags. The table rows and table data (cells) are generated when the data is dynamically pulled from the data file and passed to the edocs WebComposer object. This object formats the table rows, cells, and font characteristics of the data based on the settings defined in the Composer.

13.3.3 Naming Conventions for Charts

The Composer names each chart tag and properties file with the name of the table being charted, plus an incremental counter. For example, the first chart for the table

LocalChargeSummary would generate the chart tag

[E]LocalChargeSummary_0,U[/E] and the properties file **LocalChargeSummary_0.properties**.

If you created a second chart for the same table, the Composer would generate the chart tag **[E]LocalChargeSummary_1,U[/E]** and the properties file **LocalChargeSummary_1.properties**.

When you publish an HTML view, you will select the application name (**NatlWireless**) and specify a view name (**LocSummary**). For each chart in your HTML view, you must give the matching view name (**LocSummary**) and name the Chart view name with the chart tag (**LocalChargeSummary_0**). This name allows TBM to match each published chart properties view with the correct chart tag in the ALF.

Caution

The chart properties file overrides ALF attributes. Do not rename charts in the Composer, the ALF, or the HTML template. Use the chart properties **X.Axis.Title.String** and **X.Axis.Title.String** to define more user-friendly names for chart titles and legends.

13.3.4 About Chart Tags in the ALF

The Composer writes the chart tag and properties into the ALF, which is an XML file. In this illustration (created in Altova XML Spy), the ALF file for **NW_LocSummary.ddf** has had two charts added, **LocalLineSummary_0** and **LocalChargeSummary_0**.

XML	
DOCTYPE ALF	
ALF	
VERSION	3.0
DATA_GROUP	Local_Summary
DDF	C:\EDCSbd\samples\Nat\Wireless\NW_LocSummary.ddf
SWITCH	
HOME	Default_Template
TEMPLATES	
Template	
Name	Default_Template
SECTIONS	
CHARTS	
Chart (2)	
Name	RecordName
1 LocalLineSummary_0	LocalLineSummary
2 LocalChargeSummary_0	LocalChargeSummary
GROUPS	

The Composer creates many more default chart properties in the ALF than those you edit in the Chart dialog window. You can edit these properties directly in the ALF, or override them by editing the chart properties file. For tables of available properties and values in the ALF and in the chart properties file, see [Customizing Chart Properties](#).

13.3.5 About The Chart Properties File

The Composer also stores your chart definition in a **chart properties file**, for example *LocalLineSummary_0.properties*. This file has the same name as the table data being charted, with a counter appended. The Composer creates the properties file in the same folder as the ALF and HTML template files. You must edit these properties directly in the chart properties file. For tables of available properties and values in the ALF and in the chart properties file, see [Customizing Chart Properties](#).



Tip

You can chart the same data table in two different charts. This will increment the counter in the chart tag and properties files, for example **LocalLineSummary_0.properties** and **LocalLineSummary_1.properties**.

13.3.6 About Simulating Charts

The Composer has a Simulator tool that allows you to view your sample data as published with the current HTML template. However, Composer simulation does **not** render charts. You will need to use the Chart Simulator API, which is a command-line Java tool.

Before you simulate your chart, you will probably want to edit the ALF and properties files to get a closer first approximation of your desired chart look and feel. You can then simulate, edit, and simulate again until you are satisfied with the final design and layout.

For more information, see [Customizing Chart Properties](#), which includes procedures for [Using com.edocs.app.chart.Simulator](#).

13.4 Customizing Chart Properties

The Composer allows you to set only a few chart properties directly. To customize the format and design of your charts, you will edit chart properties in the [ALF](#) file itself; in the [chart properties](#) file; and/or in the HTML template. Any of these files can be edited with the text editor of your choice.



Caution

When you make any manual edits to ALF files, make sure to validate the XML and check to see that it is well formed.

This chapter discusses how to customize chart properties in the ALF and in the chart properties file. It also describes how to [simulate](#), or preview, charts.

13.4.1 About Chart Attributes in the ALF

The ALF, or Application Logic File, is an XML document that defines business logic and formatting for presenting statement data. An element of type ALF must contain certain required sub-elements:

```
<!ELEMENT ALF (VERSION, DATA_GROUP, DDF, SWITCH, HOME, TEMPLATES, CONTENTS,
CONDITIONS, PROFILES, BUSINESSCONDITIONS, RECORDS, PAGE_ELEMENTS, composition-
specs)>
```

Charts are defined as a subelement of the TEMPLATES element.

```
<!ELEMENT TEMPLATES (Template)+>
    <!ELEMENT Template (SECTIONS, CHARTS, GROUPS, GroupTemplate*)>
    <!ATTLIST Template
```

The CHART element in its turn defines a list of chart attributes, listed in the [Table of ALF Chart Attributes](#).

```
<!ELEMENT CHARTS (Chart*)>
<!ELEMENT Chart EMPTY>
```

```
<!ATTLIST Chart
```

Table of ALF Chart Attributes

Attribute Name	Description	Example
Name	Table name in Composer	LocalLineSummary_0
XField	X axis of chart	LocalLinePhNo
YField	Y axis of chart	LocalLineAmt
Type	Chart Type (pie, bar, &c)	Pie
HidePieLegend	Set to 1 only if Type=5 (pie)	0

Attribute Name	Description	Example
AddValueToLegend	Displays the percentage in the chart legend 1=yes, 0=no	0
Height	Total height of the canvas in pixels	300
Width	Total width of the canvas in pixels	400
HidePieLegend	Toggles the display of legends for Pie charts	
BaseURL	Points to the web application associated with the chart data. This property writes only to an existing directory, and does not create one if none exists.	
UnixChart	By default, the Composer sets UNIXChart=Pie To create other types of charts, set the chart type in the properties file.	Pie

You will notice other attributes listed in the ALF. These attributes are placed in the ALF for backward compatibility with previous versions of TBM and have no effect on the current version of TBM.

The following XML example shows the default chart properties written to **NatlWireless.alf** after creating a chart tag for the **LocalLineSummary** table.

XML Example of Chart Attributes

```
<TEMPLATES>

  <Template Name="Default_Template">

    <SECTIONS/>

    <CHARTS>

      <Chart Name="_0" RecordName="LocalLineSummary"
        TopTitle="Top Lable" BottomTitle="Bottom Lable" LeftTitle="" RightTitle=""
        XField="LocalLinePhNo" YField="LocalLineAmt" Key="0" StackedStyle="0"
        ColorScheme="0" GridLines="0" Full3D="0" AngleX="0" AngleY="0" Atribute="0"
        MarkerVolume="0" Shadow="0" MultiShape="0" Dimension_3D="0" View3DDepth="0"
        Type="1" CGITimeSpan="" BackgroundColor="White" ForgroundColor="Black"
        Height="300" Width="400" LegendShow="1" LegendToolSize="100"
        LegendToolStyle="167116800" HidePieLegend="0" SeriesColor="" LeftGap="40"
        RightGap="40" ImgQuality="75" ImgSmooth="0" AddValueToLegend="0" BaseURL="/Chart"
        UNIXChart="Pie"/>

    </CHARTS>

    <GROUPS/>

  </Template>

</TEMPLATES>
```

13.4.2 Customizing a Chart in the ALF

Adding Percentages or Values to Labels

To display the chart with data values as labels, set **AddValueToLegend=1**.

To display the chart without data values, set **AddValueToLegend=0**.

Changing Axis Titles

By default, the Composer allows you to select from the names of table rows as titles for the X- and Y-axes. Usually, these titles will not be suitable for presentation to end customers. Modify the text of axis titles in the [chart properties file](#). *Do not modify titles in the ALF as your changes will not stick.*



Caution

Title values defined in the chart properties file (**X.Axis.TitleString** and **Y.Axis.TitleString**) take precedence over those defined in the ALF (**XField** and **YField**)

Suppressing Percentage Values in Pie Charts

Pie charts (type=5) have percentage values for each slice set as the default. To suppress these values, you will need to set the URL T/F flag in the ALF for HidePieLegend attribute to 1. This **only** applies to pie charts.

13.4.3 Customizing the Chart Properties File

The chart properties file is a list of name-value pairs that control the graphic look and feel of the chart: its type, legend, labels, axes, et cetera.

- The first and most important property is [Type](#). This determines whether the data will appear as a pie, line, bar, or other type of chart. Note that *this property name and its value are both case sensitive*. All the remaining property names and their values are case insensitive.
- The naming convention of a chart property indicates its scope. For example, properties **Chart.*** affect the entire chart, while properties **X.Axis.*** affect the X-axis only. The final element of the property name indicates the property being set.
- Color and font properties have three sub-properties each. To define a color, specify individual RGB values between 0 and 255. To define a font, specify its name, style, and size.
- Most display properties are Boolean (true/false); for example, whether to display axis title or gridlines, or to display the legend vertically.
- Do not set properties that are not applicable to a chart type. For example, do not set Axis properties when requesting a Pie chart. Do not set Bar properties while rendering a Pie chart.
- For charts created using the Composer tool, the chart types: **HiLoBar**, **HorizHiLoBar**, and **Speedo** are not available, as these charts typically require additional data.

13.4.4 Chart Type

The primary chart property is **Type**, which defines the visual representation of the data. To create a pie chart, set **Type=Pie**. To create a bar chart, set **Type=Bar**.

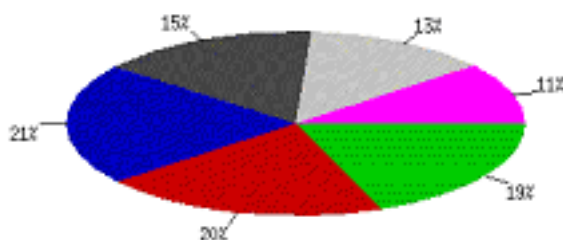
**Caution**

Both the Type property and its value are case-sensitive, unlike other chart properties in the properties file.

This section illustrates each available chart type for this example dataset.

```
X-axis label = {"Jan-Feb", "Mar-Apr", "May-Jun",
               "Jul-Aug", "Sep-Oct", "Nov-Dec"};
DataSet for 1999 = {1000.0, 1200.0, 1400, 1900.0, 1800.0, 1700.0};
DataSet for 2000 = {900.0, 1100.0, 1300, 1800.0, 1700.0, 1600.0};
X-Axis = Months
Y-Axis = Fuel Consumption;
```

13.4.5 Pie



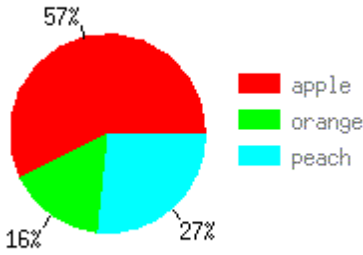
Pie chart with one slice per data point.

To define pie properties:

Pie.* properties should be defined only when **Type=Pie**. These properties control the aspect ratio (height and width) of the pie; the angle, size, and colors of the slices; and the labels and legends. For a round pie, set **Pie.Height** and **Pie.Radius** based on the height and width of the chart in pixels.

Property	Default	Description
PieLabelPosition	=2	Defines the position of the pie slice labels.
PieStartDegrees	=0	Defines the angle of the first pie slice.
PieTextLabelsOn	=false	true displays pie slice name, for example <i>College Fund</i>
PieValueLabelsOn	=false	true displays the numeric data value of each pie slice, for example 30.5
PiePercentLabelsOn	=true	true displays percent of total for each pie slice, for example 30.5%
PieLabelColor.Blue	=0	RGB value of blue (0-255).
PieLabelColor.Green	=0	RGB value of green (0-255).
PieLabelColor.Red	=0	RGB value of red (0-255).

Property	Default	Description
PieLabelFont.Name	=Times New Roman	same as java.awt.font
PieLabelFont.Style	=plain	same as java.awt.font
PieLabelFont.Size	=12	same as java.awt.font
Pie.Height	=0.5	Vertical dimension of the pie, as a percentage of plot area height. Default value produces a circle.
Pie.Width	=0.33	Horizontal dimension of the pie, as a percentage of plot area width. Default value produces a circle.
Pie.XLoc	=0.5	Horizontal center of the pie, as a percentage of plot area height.
Pie.YLoc	=0.5	Vertical center of the pie, as a percentage of plot area height.

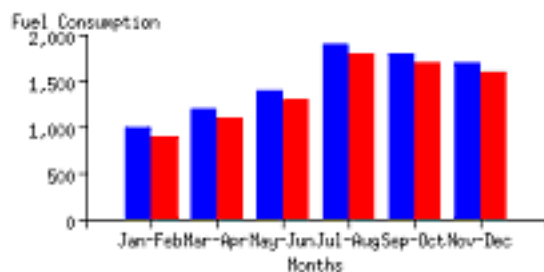
Sample Properties	Sample Chart
<pre> height=200 pixels width=300 pixels Pie.Height=0.5 (0.5 * 200=100) Pie.Width=0.33 (0.33 * 300=100) Chart.LegendVisible=true Legend.IconGap=0.02 Legend.IconHeight=0.05 Legend.IconWidth=0.07 Legend.LabelColor.Red=123 Legend.LabelColor.Blue=126 Legend.LabelColor.Green=129 Legend.LlX=0.7 Legend.LlY=0.4 Legend.VerticalLayout=false Pie.LabelPosition=2 Pie.PercentLabelsOn=true Pie.StartDegrees=0 Pie.TextLabelsOn=false Pie.ValueLabelsOn=false Pie.XLoc=0.5 Pie.YLoc=0.5 </pre>	



Tip

If labels appear too crowded, you can use a legend instead. Set **Chart.Legend.Visible=true** and specify values for legend height, width, and color. Turn off pie labels by setting **TextLabelsOn**, **ValueLabelsOn**, and **PercentLabelsOn** properties to *false*.

13.4.6 Bar

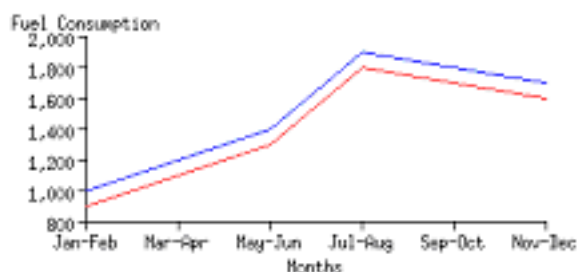


Displays each data series **vertically** in a single color (sometimes called a column chart). To display horizontally, use **HorizBar**. To display different colors for each bar in a series, use **IndBar** (horizontal) or **IndColumn** (vertical).

To define bar properties:

Property	Default	Description
Bar.Baseline	=0.0	Value from which bars will ascend or descend. Default is X-axis.
Bar.ClusterWidth	=0.8	width of a cluster of bars, as a percentage of the available space (1.0 means that clusters will touch; 0.5 means that clusters are as wide as the space separating clusters).
Bar.DoClip	=false	true will clip bar values to the outer edge of the plot area (off by default).

13.4.7 Line



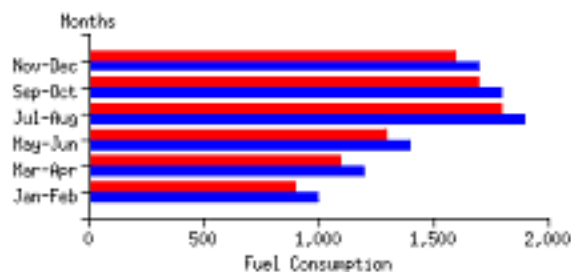
Displays data values as lines on a graph, without value labels for each data point.

To define **LineClip** property:

To clip line values at the boundary of the plot area, set **LineClip=true**. Default is **Clip=false**.

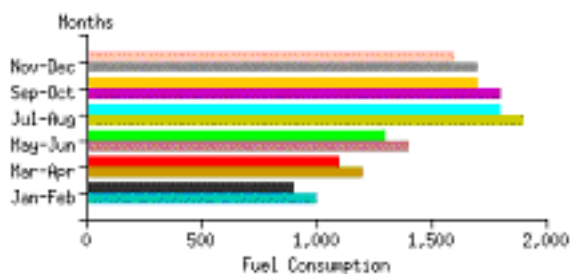
Vertical bar chart with High and Low values indicated.

13.4.8 Horizontal Bars (**HorizBar**)



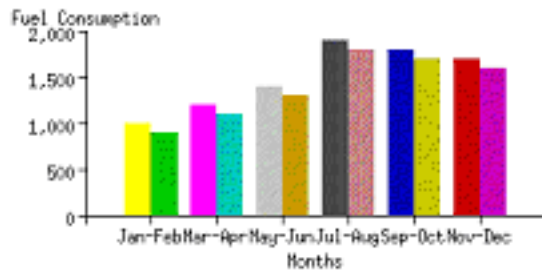
Displays each data series **horizontally** in a single color. To display vertically, (sometimes called a column chart). To display different colors for each bar in a series, use **IndBar** (horizontal) or **IndColumn** (vertical).

13.4.9 Individually Colored Bars (**IndBar**)



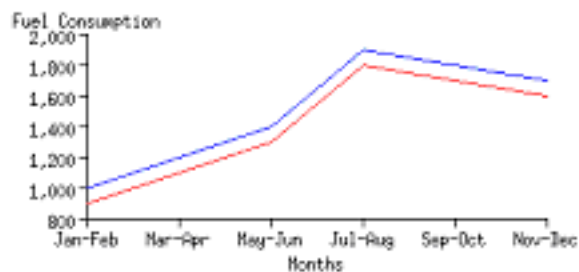
Horizontal bar chart that uses a separate color for each bar.

13.4.10 Individually Colored Columns (**IndColumn**)



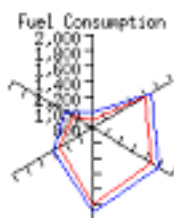
Vertical bar chart that uses a separate color for each bar.

13.4.11 (LabelLine)



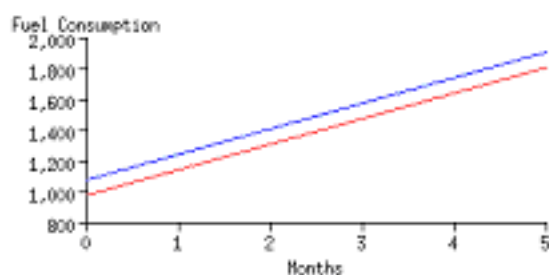
Displays data values as lines on a graph, with user-defined labels on the X-axis.

13.4.12 Polar Chart (Polar)



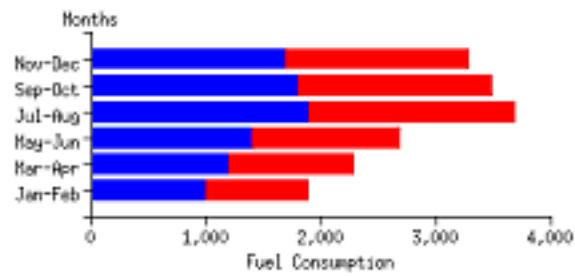
A chart that looks like a radar screen. Plots only one data value, but the scale is determined by all the data.

13.4.13 Regression (Regress)



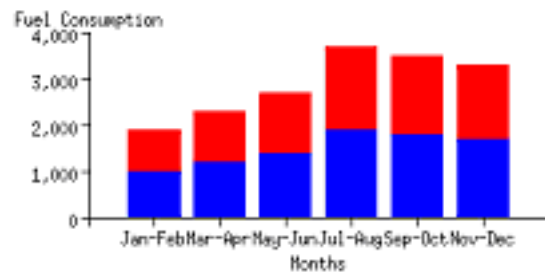
A chart that plots OLS regression for data values.

13.4.14 Stacked Bars (**StackBar**)



A chart that stacks data values horizontally.

13.4.15 Stack Column Chart (**StackColumn**)



A chart that stacks data values vertically.

13.4.16 Stick Chart (**Stick**)

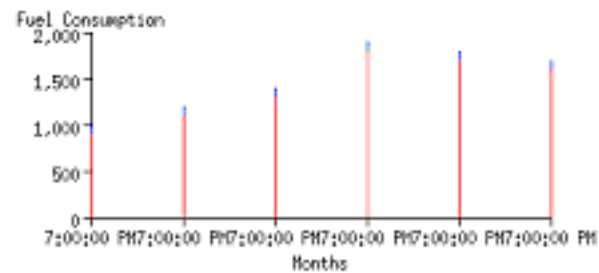


Chart that draws a vertical line to the Y-axis height of each data value.

13.4.17 StickBar Chart (**StickBar**)

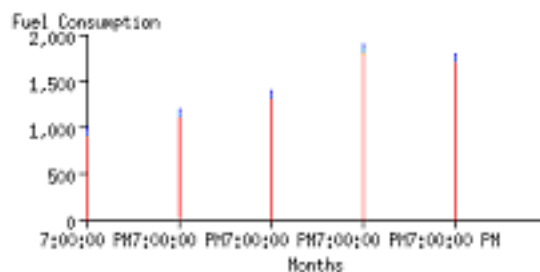


Chart that draws a vertical line to the Y-axis height of each data value.

13.4.18 General Properties

Property	Default	Description
Chart.LegendVisible	FALSE	true sets legend visible. Invisible by default.
Chart.Name	MyChart	User-defined string for chart title.
Chart.ThreeD	FALSE	true displays chart with 3D drop shadows.
Chart.XAxisVisible	TRUE	true sets X-axis visible (default).
Chart.XOffset	0	number of pixels of offset in X direction for 3D effect (default 0)
Chart.YAxisVisible	TRUE	true sets Y-axis visible (default).
Chart.YOffset	0	number of pixels of offset in Y direction for 3D effect (default 0)
ChartQuality	=1	Set to 1 for highest quality (larger) image, 0 for lower quality (smaller) image.

13.4.19 Background Properties

Titles and sub-titles are elements of the chart background. Their color, font and string value are controlled by the following properties:

Property	Default	Description
Background.Gc.FillColor.Blue	0	RGB value of blue (0-255).
Background.Gc.FillColor.Green	0	RGB value of green (0-255).
Background.Gc.FillColor.Red	0	RGB value of red (0-255).
Background.Gc.Image	<unimplemented>	UNIMPLEMENTED. Sets a background image for the chart. Do not use.
Background.Gc.LineColor.Blue	0	RGB value of blue (0-255).

Property	Default	Description
Background.Gc.LineColor.Green	0	RGB value of green (0-255).
Background.Gc.LineColor.Red	0	RGB value of red (0-255).
Background.Gc.LineWidth	1	Sets line width in pixels.
Background.Gc.MarkerColor.Blue	0	RGB value of blue (0-255).
Background.Gc.MarkerColor.Green	0	RGB value of green (0-255).
Background.Gc.MarkerColor.Red	0	RGB value of red (0-255).
Background.SubTitleColor.Blue	0	RGB value of blue (0-255).
Background.SubTitleColor.Green	0	RGB value of green (0-255).
Background.SubTitleColor.Red	0	RGB value of red (0-255).
Background.SubTitleFont.name	Times New Roman	Uses available values from java.awt.font .
Background.SubTitleFont.size	12	Uses available values from java.awt.font .
Background.SubTitleFont.style	plain	Uses available values from java.awt.font .
Background.SubTitleString	null	User-defined string for the background subtitle.
Background.TitleColor.Blue	0	RGB value of blue (0-255).
Background.TitleColor.Green	0	RGB value of green (0-255).
Background.TitleColor.Red	0	RGB value of red (0-255).
Background.TitleFont.name	Times New Roman	Uses available values from java.awt.font.
Background.TitleFont.size	12	Uses available values from java.awt.font.
Background.TitleFont.style	plain	Uses available values from java.awt.font.
Background.TitleString	null	User-defined string for the background title.

13.4.20 Plot Area Properties

The plot area is the region bounded by the axes; where the data are plotted. These properties specify the fill color for this region, and marker and grid line settings.

Property	Default	Description
Plotarea.Gc.FillColor.Blue	0	RGB value of blue (0-255).
Plotarea.Gc.FillColor.Green	0	RGB value of green (0-255).
Plotarea.Gc.FillColor.Red	0	RGB value of red (0-255).
Plotarea.Gc.LineColor.Blue	0	RGB value of blue (0-255).
Plotarea.Gc.LineColor.Green	0	RGB value of green (0-255).

Property	Default	Description
Plotarea.Gc.LineColor.Red	0	RGB value of red (0-255).
Plotarea.Gc.LineWidth	1	Sets line width in pixels.
Plotarea.Gc.MarkerColor.Blue	0	RGB value of blue (0-255).
Plotarea.Gc.MarkerColor.Green	0	RGB value of green (0-255).
Plotarea.Gc.MarkerColor.Red	0	RGB value of red (0-255).

13.4.21 Axis Properties

The axis properties control the location of the axis on the canvas, its major and minor ticks, title, grid, and labels.

- Properties listed here are for the X-axis. An identical set of properties exists for the Y-axis, **Y.Axis.***.
- Axis.Start**: By default, axes automatically determine a starting and ending value. By setting this value, you can give the axis a default minimum value. If the Axis is set to **noAutoScale**, this value will be used directly. Otherwise, this value may be adjusted slightly to yield better-looking labels. For example, if you set **X.AxisStart** to 0.01, the chart may decide to round the value down to 0.0 to create even axis increments.

Property	Default	Description
X.Axis.AutoScale	TRUE	Automatically creates X axis scale based on data values (default).
X.Axis.AxisEnd	6	Ending value of X-axis. Set to greater than or equal to number of data points anticipated.
X.Axis.AxisStart	0	Starting value on X-axis.
X.Axis.BarScaling	TRUE	Scales bars to axis length. Set true for bar charts (default).
X.Axis.GridGc.FillColor.Blue	0	RGB value of blue (0-255).
X.Axis.GridGc.FillColor.Green	0	RGB value of green (0-255).
X.Axis.GridGc.FillColor.Red	0	RGB value of red (0-255).
X.Axis.GridGc.LineColor.Blue	0	RGB value of blue (0-255).
X.Axis.GridGc.LineColor.Green	0	RGB value of green (0-255).
X.Axis.GridGc.LineColor.Red	0	RGB value of red (0-255).
X.Axis.GridGc.LineWidth	1	Sets line width in pixels.
X.Axis.GridGc.MarkerColor.Blue	0	RGB value of blue (0-255).
X.Axis.GridGc.MarkerColor.Green	0	RGB value of green (0-255).
X.Axis.GridGc.MarkerColor.Red	0	RGB value of red (0-255).

Property	Default	Description
X.Axis.GridVis	FALSE	true sets X-axis grid lines visible (invisible by default).
X.Axis.LabelAngle	0	Sets the number of degrees to rotate X axis labels.
X.Axis.LabelColor.Blue	0	RGB value of blue (0-255).
X.Axis.LabelColor.Green	0	RGB value of green (0-255).
X.Axis.LabelColor.Red	0	RGB value of red (0-255).
X.Axis.LabelFont.Name	Times Roman	Uses available values from java.awt.font.
X.Axis.LabelFont.Size	12	Uses available values from java.awt.font.
X.Axis.LabelFont.Style	plain	Uses available values from java.awt.font.
X.Axis.LabelFormat	null	Defines data format for labels, for example first three letters of month name.
X.Axis.LabelPrecision	2	Sets the number of digits past the decimal point to display X axis labels.
X.Axis.LabelVis	TRUE	true sets X axis labels visible (default).
X.Axis.LineGc.FillColor.Blue	0	RGB value of blue (0-255).
X.Axis.LineGc.FillColor.Green	0	RGB value of green (0-255).
X.Axis.LineGc.FillColor.Red	0	RGB value of red (0-255).
X.Axis.LineGc.LineColor.Blue	0	RGB value of blue (0-255).
X.Axis.LineGc.LineColor.Green	0	RGB value of green (0-255).
X.Axis.LineGc.LineColor.Red	0	RGB value of red (0-255).
X.Axis.LineGc.LineWidth	1	Sets line width in pixels.
X.Axis.LineGc.MarkerColor.Blue	0	RGB value of blue (0-255).
X.Axis.LineGc.MarkerColor.Green	0	RGB value of green (0-255).
X.Axis.LineGc.MarkerColor.Red	0	RGB value of red (0-255).
X.Axis.LineVis	TRUE	true sets X axis lines visible (default).
X.Axis.LogScaling	FALSE	true sets X axis to use log scaling; linear by default.
X.Axis.MajTickLength	5	Sets length of X axis major ticks in pixels.
X.Axis.MajTickVis	TRUE	Sets X axis major ticks visible (default).
X.Axis.MinTickLength	2	Sets length of X axis minor ticks in pixels.
X.Axis.MinTickVis	FALSE	Sets X axis minor ticks visible (default).
X.Axis.NumGrids	5	Sets the number of grid lines on the X axis to set to noAutoScale .
X.Axis.NumLabels	5	Sets the number of labels on the X axis to set to noAutoScale .

Property	Default	Description
X.Axis.NumMajTicks	5	Sets the number of major ticks on the X axis to set to noAutoScale .
X.Axis.NumMinTicks	10	Sets the number of minor ticks on the X axis to set to noAutoScale .
X.Axis.Plotarea.LIX	0.2	Shifts the horizontal position of the axis start on the canvas. Negative values shift left, positive shift right.
X.Axis.Plotarea.LIY	0.2	Shifts the vertical position of the axis start on the canvas. Negative values shift down, positive shift up.
X.Axis.Plotarea.UrX	0.8	Sets the upper right X location of the plot area as a double ranging from 0 to 1.
X.Axis.Plotarea.UrY	0.8	Sets the upper right Y location of the plot area as a double ranging from 0 to 1.
X.Axis.Side	1	
X.Axis.TickGc.FillColor.Blue	0	RGB value of blue (0-255).
X.Axis.TickGc.FillColor.Green	0	RGB value of green (0-255).
X.Axis.TickGc.FillColor.Red	0	RGB value of red (0-255).
X.Axis.TickGc.LineColor.Blue	0	RGB value of blue (0-255).
X.Axis.TickGc.LineColor.Green	0	RGB value of green (0-255).
X.Axis.TickGc.LineColor.Red	0	RGB value of red (0-255).
X.Axis.TickGc.LineWidth	1	Sets line width in pixels.
X.Axis.TickGc.MarkerColor.Blue	0	RGB value of blue (0-255).
X.Axis.TickGc.MarkerColor.Green	0	RGB value of green (0-255).
X.Axis.TickGc.MarkerColor.Red	0	RGB value of red (0-255).
X.Axis.TitleColor.Blue	0	RGB value of blue (0-255).
X.Axis.TitleColor.Green	0	RGB value of green (0-255).
X.Axis.TitleColor.Red	0	RGB value of red (0-255).
X.Axis.TitleFont.Family	Times New Roman	Uses available values from java.awt.font .
X.Axis.TitleFont.Name	Times New Roman	Uses available values from java.awt.font .
X.Axis.TitleFont.Size	12	Uses available values from java.awt.font .
X.Axis.TitleFont.Style	plain	Uses available values from java.awt.font .
X.Axis.TitleString		User-defined string for X axis title.
X.Axis.UseDisplayList	FALSE	UNIMPLEMENTED. Retrieves objects using mouse click events. Do not use.

13.4.22 Legend Properties

Property	Default	Description
Legend.BackgroundGC.Gc.FillColor.Blue	0	RGB value of blue (0-255).
Legend.BackgroundGC.Gc.FillColor.Green	0	RGB value of green (0-255).
Legend.BackgroundGC.Gc.FillColor.Red	0	RGB value of red (0-255).
Legend.BackgroundGC.Gc.Image	<unimplemented>	Sets a background image for the legend. Do not use.
Legend.BackgroundGC.Gc.LineColor.Blue	0	RGB value of blue (0-255).
Legend.BackgroundGC.Gc.LineColor.Green	0	RGB value of green (0-255).
Legend.BackgroundGC.Gc.LineColor.Red	0	RGB value of red (0-255).
Legend.BackgroundGC.Gc.LineWidth	1	Sets line width in pixels.
Legend.BackgroundGC.Gc.MarkerColor.Blue	0	RGB value of blue (0-255).
Legend.BackgroundGC.Gc.MarkerColor.Green	0	RGB value of green (0-255).
Legend.BackgroundGC.Gc.MarkerColor.Red	0	RGB value of red (0-255).
Legend.BackgroundVisible	TRUE	Set false to avoid displaying background in chart legend.
Legend.IconGap	0.02	separation between rows of the legend.
Legend.IconHeight	0.05	Legend icon height $0 < k < 1.0$, where 1.0 = full height of canvas
Legend.IconWidth	0.07	Legend icon width $0 < k < 1.0$, where 1.0 = full width of canvas
Legend.LabelColor.Blue	0	RGB value of blue (0-255).
Legend.LabelColor.Green	0	RGB value of green (0-255).
Legend.LabelColor.Red	0	RGB value of red (0-255).
Legend.LabelFont.Name	Times New Roman	Uses available values from java.awt.font .
Legend.LabelFont.Size	12	Uses available values from java.awt.font .
Legend.LabelFont.Style	plain	Uses available values from java.awt.font .
Legend.LIX	0	lower x left corner $0 < y < 1.0$, 1.0 = full width
Legend.LIY	0	lower y left corner $0 < y < 1.0$, 1.0 = full height

Property	Default	Description
Legend.UrX	0	upper x right corner $0 < y < 1.0$, $1.0 =$ full width
Legend.UrY	0	upper y right corner $0 < y < 1.0$, $1.0 =$ full height
Legend.VerticalLayout	FALSE	Set true to display legend at side of chart, false to display below chart.

13.4.23 Favorite Colors Properties

You can specify **favorite colors** as RGB values, to fill pie slices, bars, and other data values. For example, you may want to match the chart colors to the color scheme of the embedding page. Favorite colors are specified using the prefix **Favorite.1.Color**, where 1 is the first color in the series.

- **Number:** Favorite colors become active when the number of colors defined is greater than or equal to the number of data points displayed (unless the for individual colors). For example, if a Pie has six slices but only five favorite colors specified, the favorite colors will not be used. This is because there is no way to guess which colors would go well with those already specified.
- **Order:** The favorite colors are used in the order specified. Define each color to be distinguishable from adjacent colors for contrast and readability.

This example defines two favorite colors:

```
Favorite.1.Color.Red=201
Favorite.1.Color.Blue=92
Favorite.1.Color.Green=132

Favorite.2.Color.Red=51
Favorite.2.Color.Blue=52
Favorite.2.Color.Green=53
```

13.4.24 Default Chart Properties

The following properties are the default values created in the chart properties file. For a full listing, see [Appendix A: Default Chart Properties and ALF Attributes](#).

```
Type=Pie
Legend.BackgroundVisible=true
Legend.IconGap=0.02
Legend.IconHeight=0.05
Legend.IconWidth=0.07
Legend.LabelColor.Red=0
Legend.LabelColor.Blue=0
Legend.LabelColor.Green=0
Legend.LabelFont.Name=Times New Roman
Legend.LabelFont.Style=plain
Legend.LabelFont.Size=12
Legend.LlX=0.0
```

```

Legend.LlY=0.0
Legend.UrX=0.0
Legend.UrY=0.0
Legend.VerticalLayout=false
X.Axis.CullingLabel=false;
X.Axis.AutoScale=true
X.Axis.AxisEnd=6.0
X.Axis.AxisStart=0.0
X.Axis.BarScaling=true
X.Axis.GridVis=false
X.Axis.LabelAngle=0
X.Axis.LabelColor.Red=0
X.Axis.LabelColor.Blue=0
X.Axis.LabelColor.Green=0
X.Axis.LabelFont.Name=Times New Roman
X.Axis.LabelFont.Style=plain
X.Axis.LabelFont.Size=12
X.Axis.LabelFormat=null
X.Axis.LabelPrecision=2
X.Axis.LabelVis=true
X.Axis.LineVis=true
X.Axis.LogScaling=false
X.Axis.MajTickLength=5
X.Axis.MajTickVis=true
X.Axis.MinTickLength=2
X.Axis.MinTickVis=false
X.Axis.NumGrids=5
X.Axis.NumLabels=5
X.Axis.NumMajTicks=5
X.Axis.NumMinTicks=10
X.Axis.Plotarea.LlX=0.2
X.Axis.Plotarea.LlY=0.2
X.Axis.Plotarea.UrX=0.8
X.Axis.Plotarea.UrY=0.8
X.Axis.TitleRotated=true
X.Axis.TitleColor.Red=0
X.Axis.TitleColor.Blue=0
X.Axis.TitleColor.Green=0
X.Axis.TitleFont.Family=TimesNewRoman
X.Axis.TitleFont.Name=Times New Roman
X.Axis.TitleFont.Style=plain
X.Axis.TitleFont.Size=12
X.Axis.TitleString=RemembertosetXaxistitle!

```

```

X.Axis.UseDisplayList=false
Y.Axis.CullingLabel=false;
Y.Axis.AutoScale=true
Y.Axis.AxisEnd=6.0
Y.Axis.AxisStart=0.0
Y.Axis.BarScaling=true
Y.Axis.GridVis=false
Y.Axis.LabelAngle=0
Y.Axis.LabelColor.Red=0
Y.Axis.LabelColor.Blue=0
Y.Axis.LabelColor.Green=0
Y.Axis.LabelFont.Name=Times New Roman
Y.Axis.LabelFont.Style=plain
Y.Axis.LabelFont.Size=12
Y.Axis.LabelFormat=null
Y.Axis.LabelPrecision=2
Y.Axis.LabelVis=true
Y.Axis.LineVis=true
Y.Axis.LogScaling=false
Y.Axis.MajTickLength=5
Y.Axis.MajTickVis=true
Y.Axis.MinTickLength=2
Y.Axis.MinTickVis=false
Y.Axis.NumGrids=5
Y.Axis.NumLabels=5
Y.Axis.NumMajTicks=5
Y.Axis.NumMinTicks=10
Y.Axis.Plotarea.LlX=0.2
Y.Axis.Plotarea.LlY=0.2
Y.Axis.Plotarea.UrX=0.8
Y.Axis.Plotarea.UrY=0.8
X.Axis.TitleRotated=true
Y.Axis.TitleColor.Red=0
Y.Axis.TitleColor.Blue=0
Y.Axis.TitleColor.Green=0
Y.Axis.TitleFont.Family=TimesNewRoman
Y.Axis.TitleFont.Name=Times New Roman
Y.Axis.TitleFont.Style=plain
Y.Axis.TitleFont.Size=12
Y.Axis.TitleString=RemembertosetXaxistitle!
Y.Axis.UseDisplayList=false
Pie.LabelColor.Red=0
Pie.LabelColor.Blue=0

```

```

Pie.LabelColor.Green=0
Pie.LabelFont.Name=Times New Roman
Pie.LabelFont.Style=plain
PieLabelFont.Size=12
Pie.LabelPosition=2
Pie.PercentLabelsOn=true
Pie.StartDegrees=0
Pie.TextLabelsOn=false
Pie.ValueLabelsOn=false
Pie.Height=0.6
Pie.Width=0.6
Pie.XLoc=0.5
Pie.YLoc=0.5
Chart.LegendVisible=false
Chart.Name=MyChart
Chart.ThreeD=false
Chart.XOffset=0
Chart.YOffset=0
Chart.YAxisVisible=true
Chart.XAxisVisible=true
Chart.Quality=1.0

```

13.4.25 Customizing Default Properties

The default settings for the chart properties file are stored in the text file **ChartDefaults.properties** inside the **com.edocs.app.chart** directory of **edx_servlet.jar**. If you are creating a series of similar charts, you may find it convenient to modify the default properties as a base template.

The **eaSample** web application contains two instances of **edx_servlet.jar**: one in the **WEB-INF/lib** directory of the EAR file **ear-easample-ear** and another in the **WEB-INF/lib** directory of WAR file **war-easample.war**. You can edit either instance of **ChartDefaults.properties** as long as you add the edited version of **edx_servlet.jar** to your classpath.

Unjar the EAR and WAR archive files, and then unjar **edx_servlet.jar**, to find **ChartDefaults.properties** in each archive. For example, the path to the default properties file in the WAR file for a default installation of TBM (WebLogic for Windows 2000) is:

```

C:/EDCSbd/samples/eaSample/J2EEApps/weblogic/ear-easample/war-easample/WEB-INF/lib/edx_servlet/ com/edocs/app/chart/ChartDefaults.properties

```

Open **ChartDefaults.properties** with a text editor and make any desired changes to the default properties; for example, change the default if you are creating a series of bar charts. Jar up the servlet, WAR, and EAR files, and add your modified **edx_servlet.jar** to your classpath. Now, when you create a chart in the Composer, the default chart properties should reflect your new settings.

13.4.26 Previewing Charts with com.edocs.app.chart.Simulator

The Chart Simulator API is a simple command line interface for `com.edocs.app.chart.ChartClient.java`. The API `chart.Simulator` requires a Java environment with `javachart.jar`, `edx_servlet.jar` and `edx_client.jar` in the default classpath. These JAR files are installed with TBM, but *you must add them to your classpath* to use the Simulator.

Setting the Display Environment for Simulation

You can control where your charts are simulated by setting your [display awareness](#) to either your production or deployment server. For simple previews to check if data is being correctly retrieved, a local simulation on your production workstation may be fine. However, remember that the appearance of your chart is controlled by the **deployment server**. This server may have different fonts available or be running a virtual display with different resolution or other graphics settings. Always preview your charts in a deployment environment before finalizing your chart properties and ALF.

For more information on display environments, including display devices, permissions, and awareness, see [Configuring Charting For Your Server](#).

Formatting Data Strings for Chart Data

The Charting API passes chart data directly as one or more encoded data strings, for example:

```
"Series1*F*30*apple*56.8*orange*12.5*banana"
```

Enclose the data string in quotes, and separate each value with an asterisk (*). The first value in the string must be the data series name. The second is a T/F value that determines whether to label data values with the pie slice percentage (T=labels, F=no labels).

All remaining values in the string must be value/name pairs, where the first item in each pair is the numeric data value and the second item is its label name.



Tip

Remember to put values first, then labels (the opposite of a standard name/value pair).

The Simulator also takes parameters for the height and width of the chart canvas. If you find that labels or legends are clipped or cut off, adjust your chart canvas and properties file settings to accommodate the maximum length of legends and other objects.

To simulate a chart:

1. Edit your chart properties file.
2. Add `javachart.jar`, `edx_servlet.jar` and `edx_client.jar` to your classpath.
3. (optional) Create a text file with your formatted data strings, which you can then copy and paste into the command.
4. Run the Simulator from the command line.

```
java com.edocs.app.chart.Simulator propsFileName imgOutputFilename.jpg width
height encodedDataString1 encodedDataString2
```

5. View the generated image file in your browser. By default, the generated image is saved in the directory containing the ALF and properties file.

The Composer can simulate only charts containing a single data string. To display multiple datasets, you must customize the charting servlet to extract and present data as multiple strings. For more information, see your edocs Professional Services representative.

Parameters

Parameter	Description
propsFileName	Chart properties file
imgOutputFilename	File name for image output. JPEG format required
Width	Width of chart canvas in pixels.
Height	Height of chart canvas in pixels.
encodedDataString	A single data string enclosed in quotes and separated by asterisks. See Formatting Data Strings for the Simulator .

13.5 Publishing Charts

The TBM application server compiles charts dynamically at web time. When an HTML template contains a chart tag, the web application requests the table from the TBM database and posts an HTTP request to the charting servlet. The servlet uses the chart properties file published for that version set to format the data from the table, and renders the chart as a JPEG embedded in the dynamic HTML.

13.5.1 Before Publishing Charts

To prepare your application for chart viewing:

1. Create an application, in this example **NatlWireless**.
2. Create, configure, and run an Indexer job to index your dataset.
3. Publish an HTML web view, in this example

edocs

Create a version set for HTML

Application:

View Type:

View Name:

DDF File:

ALF File :

HTML Template:

13.5.2 Publishing a Chart View

You must publish a CHART view for *each individual chart* in an online statement. Chart views merge the DDF and ALF information from the specified HTML web view with the properties you defined for the chart, and embed the chart data in the Chart URL published by the HTML web view.



Caution

If you publish an HTML web view for an ALF with a chart tag, you must publish a CHART view for that chart before it will display. If there is no chart view available, the HTML web view will display only a placeholder for the chart graphic.

To publish a Chart view:

1. In the Command Center, select Publisher from the navigation bar. A new Publisher window appears.
2. Select Create from the navigation bar. The Create a version set for CHART window appears.

The screenshot shows a web application titled "edocs" with a sidebar menu containing "Browse", "Create", "Fetch", "Delete", and "Help". The main content area is titled "Create a version set for CHART". It contains the following fields and buttons:

- Application:** A dropdown menu with "NatlWireless" selected.
- View Type:** A text field containing "CHART".
- View Name:** An empty text field.
- Properties File:** A text field containing "C:\EDCSbd\samples\NatlWireles" followed by a "Browse..." button.
- Buttons:** "Submit", "Clear", and "Help" buttons are located below the Properties File field.
- Footer:** Copyright notice: "© Copyright 1997-2002 edocs®, Inc. All Rights Reserved. edocs is Reg. U.S. Pat. & Tm. Off. [Privacy Policy](#)".

3. Select the name of your application from the dropdown list, for example, NatlWireless. The view type is set to CHART.
4. Enter the View Name associated with this chart. This view name must be the name of the chart, for example LocChargeSummary_0.
5. Enter the path to the chart properties file, for example
`C:/EDCSbd/samples/NatlWireless/LocChargeSummary_0.properties`
 This file must have an extension of ***.properties**.
6. If you have modified the view name since creating your ALF, update the ALF to point to the correct web view for your application,
7. Click Submit to publish your chart view.

13.5.3 Viewing Charts in Statements

To view charts in statements:

1. Index your statement data with an Indexer job.
2. Publish at least one HTML web view with an ALF containing chart tag(s).
3. Publish a Chart view for each chart.
4. Browse your web application and enroll one or more customers.
5. Log in as the customer whose statement you wish to view and browse the statement with the chart(s).

13.6 Designing Custom Charts with the Charting Servlet

The previous sections describe how to use the edocs charting servlet to compose and publish charts in online statements. The `com.edocs.app.chart` API allows you to create your own charting servlets to generate customized charts. Your servlet will be creating an instance of the `ChartClient` class.



Caution

Ensure that you set servlet response type appropriately before sending any output to the servlet. The response object of the servlet is a required input to the `generateChart` method of the `ChartClient` class, which streams the chart as a jpeg. Always set `response.setContentType("image/jpeg")` in custom servlets.

Browse to the charting servlet with a URL of this syntax for your application:

```
http://hostname:portNum/eStatements/Chart?app=Charter\
&ddn=MyApp&viewName=MyView&W=200&H=300\ &data=encodedData.
```

`ChartData` is a constructor that takes an encoded data string. A chart may have one or more such encoded data sets.

The width and height parameters are not present in the properties file, since the dimensions of the canvas are not actual properties of the chart.



Tip

Remember to register any custom servlets you create in the web.xml file for your web application.

13.6.1 Customizing Charter.java

This topic presents the complete code for the default charting servlet that ships with TBM, with comments on where and how to begin customization.

To customize your servlet, you will need to import the following packages, as well as any other packages you intend to use. Package `chart` is the Charting API. Class `App` is the base class for all TBM application servlets, and class `LoginRequired` is the interface which signals that an account is required before access should be granted.

```
package com.edocs.app.chart;

import java.io.*;
import java.util.*;
import java.awt.Graphics;
import java.awt.Image;
import java.awt.Frame;
import javax.servlet.*;
import javax.servlet.http.*;
import com.edocs.app.App;
import com.edocs.app.LoginRequired;
```

Your custom servlet can extend the base servlet class with a new instance of the **Charter** class, which contains the **getDataSets** method that requests the **ChartData** object. This class takes the response from the client browser and sends it to the application server, which in turn fulfills the servlet request and passes a chart URL back to the browser. *You may customize a chart servlet to obtain its data from another source, or in a different format.*

```
public class Charter extends App implements LoginRequired {
    static private boolean DEBUG = Constants.DEBUG;

    public void doPost(HttpServletRequest request,
        HttpServletResponse response) {
```

The charting engine returns images in JPEG format. You **must** set the content type of your servlet to JPEG for the output stream.

```
        try {
            response.setContentType("image/jpeg");
            ServletOutputStream out = response.getOutputStream();
```

You will then request the parameters from the DDN (data source) and the CHART view name (ALF for presentation logic and properties from the chart properties file). Together, these parameters identify the file to retrieve from the versioning system and determine the chart properties. If you have multiple charts in a single statement, create a

```
String ddn = request.getParameter("ddn");
String viewName = request.getParameter("viewName");
```

Requesting the width and height parameters will determine the canvas size of your chart. You set these parameters in the chart properties file.

```
int width = Integer.parseInt(request.getParameter("W"));
int height = Integer.parseInt(request.getParameter("H"));
```

You then call a Java Properties object that loads the specified chart properties, DDN, and view name with the **getChartPropsStream** method of the **PublisherWrapper** class. This links the data source and graphic elements of the chart.

```
Properties chartProps = new Properties();
chartProps.load(PublisherWrapper.getChartPropsStream(ddn,viewName));
```

The **ChartData** class constructs the datasets for the chart from the encoded data passed in the chart URL. For details of this constructor, which has five signatures, see [Class ChartData](#) in [Application Programming Interfaces \(APIs\) for Charting](#).

```
ChartData[] dataSets = getDataSets(request);
```



Tip

You can insert a custom error message here, for example to advise of too much data in the chart URL. See [Example: Custom Error Message](#).

Your servlet now creates a new **ChartClient** to hold the chart properties and the dimensions of the canvas, and generates the chart as an **out** object for the servlet response, catching any exceptions.

```
ChartClient cl = new ChartClient(chartProps, width, height);
cl.generateChart(out, dataSets);
```

```

        } catch (Exception e) {
            e.printStackTrace();
            doForwardException(request, response, e);
        }
    }

    public void doGet(HttpServletRequest request, HttpServletResponse response)
    {
        doPost(request, response);
    }

```

ChartData gets the encoded data string from the chart object in the ALF. **Any data properties specified in the chart properties file will override these ALF attributes.**

```

static private ChartData[] getDataSets(HttpServletRequest request)
    throws ChartException {
    String[] dataStrs = request.getParameterValues("data");
    int num = dataStrs.length;
    ChartData[] dataSets = new ChartData[num];
    for (int i = 0; i < num; i++) {
        dataSets[i] = new ChartData(dataStrs[i]);
        if (DEBUG) {
            System.out.println("DataSet(" + i + ") : " + dataSets[i]);
        }
    }
    return dataSets;
}

```

Example: Custom Error Message

```

chartProps.load(PublisherCommon.getChartPropsStream(ddn,viewName));
try {
    ChartData[] dataSets = getDataSets(request);
} catch (Exception be) {
    String msg = e.getMessage(); if (msg.indexOf("data format error")) { // then
    perhaps our URL is too long // get the sorry too much charting data // open the
    gif tooMuch.gif // write it to the output stream return; } else { throw be; }
}

ChartClient cl = new ChartClient(chartProps, width, height);
cl.generateChart(out, dataSets);

} catch (Exception e) { e.printStackTrace(); doForwardException(request,
response, e); } }

```

13.7 Troubleshooting Charts

13.7.1 Charting Checklist

- Are xwindow displays enabled on the web-server machine? In an x-term window or a terminal that knows a notion of "DISPLAY" enter "echo \$DISPLAY." If you get a non-null string, run `xhost +` on the machine indicated in the display variable.
- Does your machine have a physical display device or is it headless? A machine without a physical display requires an x-virtual frame buffer, such as `xvfb`.
- Does your web/application server know where to send its x-displays? Make sure the `DISPLAY` environment variable is correctly set, either in the start script for your web/application server or in the xterm for your start command.
- Is `xhost` running? Ensure you have not closed the xterm which issued "`xhost +`" unless you have "`xhost +`" as part of your server startup script.
- Have you published a CHART view in the TBM Publisher? A Chart view requires a chart properties file; make sure you have published the one associated with your TBM application.
- Does the BaseURL charting attribute in the ALF file match your web application name? This attribute points the servlet to the correct CHART view. Make sure they match.
- Can you see charts in statements? If not, repeat the steps above. If you are still having trouble:
- How long is your chart URL? For large datasets, you may need to customize the charting servlet.
- Does your data contain special characters? The chart servlet may not handle these characters correctly. You will need to customize the charting servlet.

13.7.2 Common Problems and Known Issues

AIX Does Not Display Charts

The X Windows client for AIX systems requires the X11 package, which comes with the O/S but is not installed by default. To check whether X11 is installed, run `smitt` and check the installed packages option for AIX Windows X11 libraries, or look in the default directory `/usr/lpp/X11`.

Pie Chart Displays When Another Chart Type Is Selected

The Composer's chart dialog box offers three, , and Pie. However, the Composer will create only a Pie chart by default. To change the chart type and display an example, edit the chart properties file (not the ALF) and simulate the chart with the Simulator API.

Small Segments Collide In Pie Charts

Remove the % values that appear close to the pie, by setting chart properties

```
Pie.PercentLabelsOn=false
```

```
Pie.TextLabelsOn=false
```

UNDO Button In Composer Does Not Affect ALF

Inserting a chart modifies both the ALF and the HTML template, enabling the UNDO button. If you then click UNDO, only the HTML template changes are reversed—not those in the ALF. This can cause the Composer to fail when processing a section of a statement. Use caution in using the UNDO button to reverse changes.

Chart Quality Is Poor

The default chart property is `Chart.Quality=.75`. For highest quality charts, set chart property `Chart.Quality=1`.

ALF Axis Titles Overwritten By Properties File

Title values defined in the chart properties file (`X.Axis.TitleString` and `Y.Axis.TitleString`) take precedence over those defined in the ALF (`XField` and `YField`).

Changing Addtolegend In ALF Does Not Change Chart URL

Changing the value in the ALF for the property `AddValueToLegend` does not change the URL "T"/"F" property. Instead, it actually passes the value as part of the legend. In order to change the URL "T"/"F" property through the ALF file you will need to set the `HidePieLegend` property to 1. This will only work if the Type property is set to 5 for pie.

Title Fonts Do Not Appear Bold

Setting font properties, for example `Y.Axis.TitleFont.Style=Bold`, on a headless server requires that fonts be available **and** requires a virtual display, or virtual frame buffer, such as `xvfb`. To display fonts and styles correctly, see [Display Devices and xvfb](#) and Configuring a Headless Server for Charting.

Bold Italic Does Not Display Correctly

Setting fonts to both bold and italic in the chart properties file may cause text to display as a bitmap. Charting implements fonts through `java.awt.font`, and the bold italic combination is handled as a bitmap of `java.awt.Font.BOLD` and `java.awt.ITALIC`.

Chart Servlet Suppresses Commas And Spaces

In a legend label, the charting servlet interprets "July 25, 2002" as "July252002." The workaround is to reformat the data at the JSP layer, but this does not work for web views.

13.8 Application Programming Interfaces (APIs) for Charting

13.8.1 Package com.edocs.app.chart Description

Contains classes, constructors, and methods to render and publish charts as JPEG graphics and to extend the charting servlet (an instance of the Charter class).

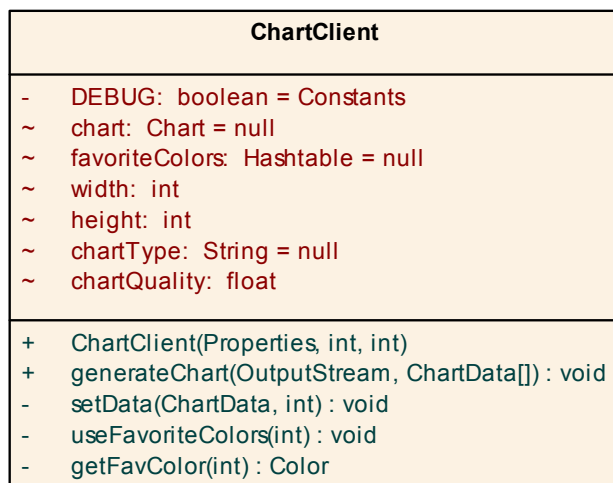
13.8.2 Class ChartClient

Contains a constructor and methods to draw a chart as a JPEG graphic.

Available Chart Types

Type	Description
Pie	Pie chart with one slice per data point.
Bar	Displays each data series vertically in a single color (sometimes called a column chart). To display horizontally, use HorizBar . To display different colors for each bar use IndBar (horizontal) or IndColumn (vertical).
Line	Displays data values as lines on a graph, without value labels for each data point.
HiLoBar	Vertical bar chart with High and Low values indicated.
HorizBar	Displays each data series horizontally in a single color. To display vertically, use Bar (sometimes called a column chart). To display different colors for each bar, use IndBar (horizontal) or IndColumn (vertical).
HorizHiLoBar	Horizontal bar chart with High and Low values indicated.
IndBar	Horizontal bar chart that uses separate color for each bar.
IndColumn	Vertical bar chart that uses a separate color for each bar.
LabelLine	Displays data values as lines on a graph, with user-defined labels on the X-axis.
Polar	A chart that looks like a radar screen. Plots only one data value, but the scale is determined by all the data.
Regress	Subclass of Line chart that plots OLS regression for data values.
Speedo	A chart that looks like a gauge or speedometer, similar to Polar.
StackBar	Bar chart that stacks data values horizontally.
StackColumn	Bar chart that stacks data values vertically.
Stick	Chart that draws a vertical line to the Y-axis height of each data value.
StickBar	Chart that draws a vertical line to the Y-axis height of each data value.

Class Diagram



Constructors

`chartClient(java.util.Properties properties, int canvasWidth, int canvasHeight)`
throws `ChartException`

The `ChartClient` constructor takes a java properties object that specifies the default properties for the chart, and integers that specify the dimensions of the canvas in pixels.

Parameters

Parameter	Description
properties	Default property list. See <code>java.util.Properties</code> .
canvasWidth	Integer specifying the chart width in pixels.
canvasHeight	Integer specifying the chart height in pixels.

Methods

`generateChart(java.io.OutputStream out, ChartData[] dataSets)` throws
`ChartException`

generateChart is invoked to render the chart as a graphic.

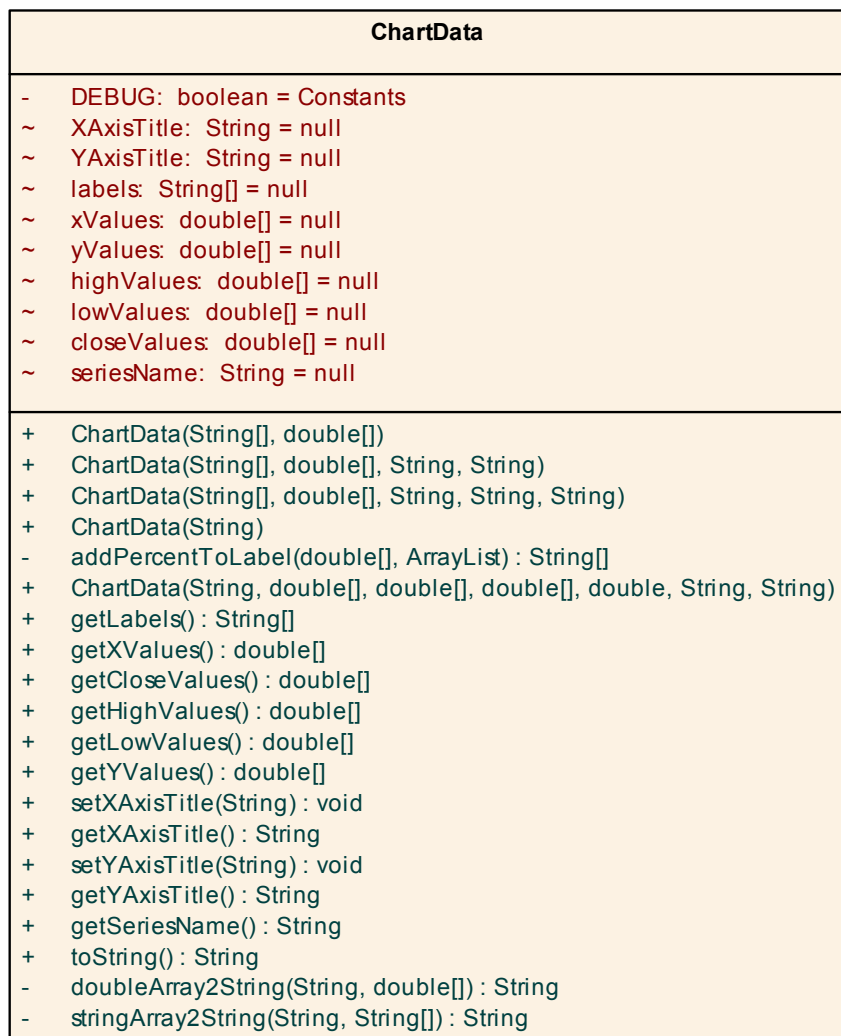
Parameters

Parameter	Description
out	Defines an output stream, for example to generate the output for a servlet response object.
data	String specifying the data to be charted.

13.8.3 Class ChartData

Contains a constructor and methods to create the **ChartData** object required by the **ChartClient**. Also contains **get** and **set** methods for constructor parameters. For details, see the *TBM API Specification*.

Class Diagram



Constructors

Five signatures of **ChartData** construct the **ChartData** object to pass to the **ChartClient**.

```
ChartData(java.lang.String URLEncodedDataStr)
```

Constructs an object containing an encoded URL in an HTTP Get request, or any raw data string.

```
ChartData(java.lang.String[] theLabels, double[] vals)
```

Constructs an object containing chart labels and name-value pairs of data values, as from a properties file.

```
ChartData(java.lang.String[] theLabels, double[] vals,
java.lang.String xTitle, java.lang.String yTitle)
```

Constructs an object containing chart labels, name-value pairs of data values, and axis titles, as from a properties file. Use when charting a single data series.

```
ChartData(java.lang.String[] theLabels, double[] vals, java.lang.String xTitle,
java.lang.String yTitle, java.lang.String dataSeriesName)
```

Constructs an object containing chart labels, name-value pairs of data values, axis titles, and the name of each data series, as from a properties file. Use when charting multiple data series, as for stacked lines or bars.

```
ChartData(java.lang.String dataSeriesName, double[] xVals, double[] hiVals,
double[] loVals, double[] closeVals, java.lang.String xTitle, java.lang.String
yTitle)
```

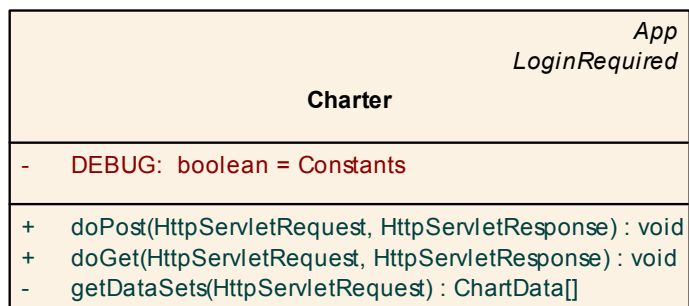
Constructs an object containing parameters for high-low bar charts.

Parameters

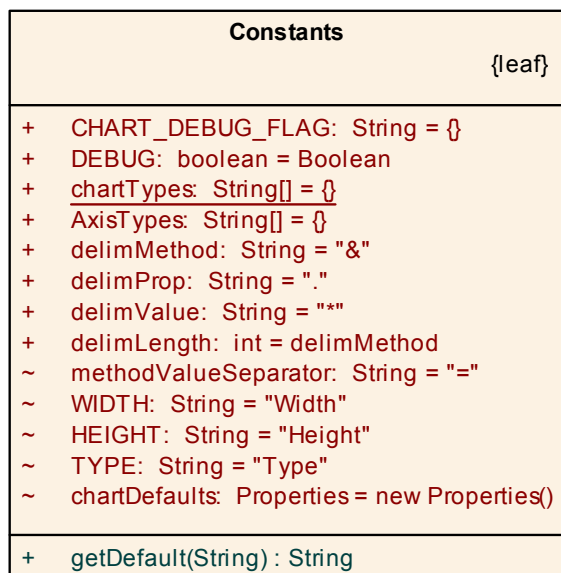
Parameter	Description
closeVals[]	Double parameter for closing values in a high-low bar chart.
dataSeriesName	Display name for the data series being charted. Use when displaying multiple data series in a single chart.
hiVals	Double parameter for high values in a high-low bar chart.
loVals	Double parameter for low values in a high-low bar chart.
theLabels	String array containing values for data labels.
URLEncodedDataStr	Chart data passed as a string. For syntax, see Formatting Data Strings for the Simulator .
vals	Array of doubles, the value to chart
xTitle	Display name for the X-Axis.
xVals	double[]
yTitle	Display name for the Y-Axis.

13.8.4 Class Charter

Servlet class for the charting servlet. Contains the `getDataSets` method that requests the `ChartData` object. Implements `com.edocs.app.LoginRequired`, `javax.servlet.Servlet`, and extends `com.edocs.app.App`. Also contains `doPost` and `doGet` methods that override those in class `com.edocs.app.App`. For details, see the *TBM API Specification*.

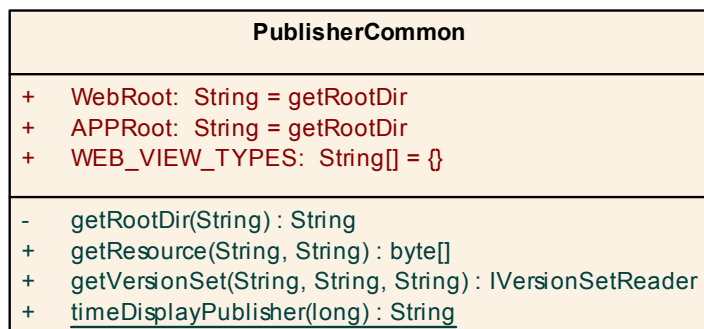
Class Diagram**13.8.5 Class Constants**

Contains a constructor and default fields for defining custom chart parameters. For details, see the *TBM API Specification*.

Class Diagram**13.8.6 Class PublisherCommon**

Contains a constructor, fields, and methods to retrieve the root directory and web views for the chart from the Publisher. For details, see the *TBM API Specification*.

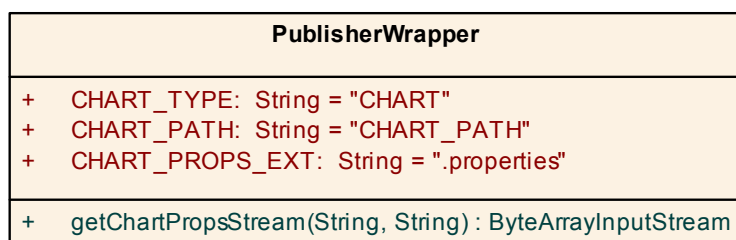
Class Diagram



13.8.7 Class PublisherWrapper

Contains a constructor, fields, and methods to retrieve chart properties as an input stream. For details, see the *TBM API Specification*.

Class Diagram



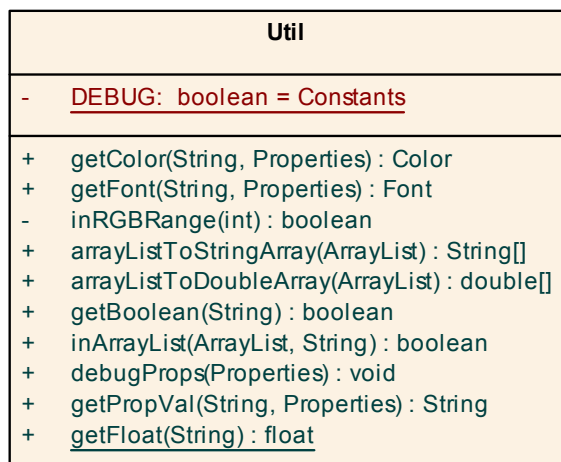
13.8.8 Class Simulator

Constructor and methods that behave like a shell command to create a JPEG image of the specified data and chart properties. Extends class `java.awt.Frame`. For usage, see [Previewing Charts with com.edocs.app.chart.Simulator](#). For details, see the *TBM API Specification*.

13.8.9 Class Util

Utility class to support the Charting API. For details, see the *TBM API Specification*.

Class Diagram



13.9 Default Properties and Attributes

13.9.1 ChartDefaults.properties

```
#####
# Kavachart3.2 properties influencing chart presentation # initially 3.1,
# with addition of CullingLabel, its 3.2
#
#####

# -----
# Primary Properties
# -----

# Chart type: Pie

# DateLine
# FinCom
# HiLoBar
# HiLoClose
# HorizBar == to generate horizontal bar chart
#
# **** Please note, the property name, "Type" IS case sensitive, and its
# **** value is case sensitive.
Type=Pie

# -----
```

```

# Legend related
# -----

Legend.BackgroundVisible=true
## -- separation between rows of the legend
Legend.IconGap=0.02
## -- Legend icon height 0 < k < 1.0, where 1.0 = full height of canvas
Legend.IconHeight=0.05
## -- Legend icon width 0 < k < 1.0, where 1.0 = full width of canvas
Legend.IconWidth=0.07
## -- R G B values
Legend.LabelColor.Red=0
Legend.LabelColor.Blue=0
Legend.LabelColor.Green=0
## -- this needs to be broken up into its components
Legend.LabelFont.Name=Times New Roman
Legend.LabelFont.Style=plain
Legend.LabelFont.Size=12

## -- lower x left corner 0 < y < 1.0, 1.0 = full width
Legend.LlX=0.0
## -- lower y left corner 0 < y < 1.0, 1.0 = full height
Legend.LlY=0.0
## -- upper x right corner 0 < y < 1.0, 1.0 = full width
Legend.UrX=0.0
## --upper y right corner 0 < y < 1.0, 1.0 = full height
Legend.UrY=0.0
## -- legend below chart
Legend.VerticalLayout=false

## -- To change the Legend's Graphic component
## Legend.BackgroundGC.Gc.FillColor.Red=0
## Legend.BackgroundGC.Gc.FillColor.Blue=0
## Legend.BackgroundGC.Gc.FillColor.Green=0
## Legend.BackgroundGC.Gc.LineColor.Red=0
## Legend.BackgroundGC.Gc.LineColor.Blue=0
## Legend.BackgroundGC.Gc.LineColor.Green=0
## Legend.BackgroundGC.Gc.LineWidth=1
## Legend.BackgroundGC.Gc.MarkerColor.Red=0
## Legend.BackgroundGC.Gc.MarkerColor.Blue=0
## Legend.BackgroundGC.Gc.MarkerColor.Green=0
## Legend.BackgroundGC.Gc.Image -- unimplemented

```

```

# -----
# Class name = javachart.chart.Axis
# Specific to X axis
# -----
## to skip labels that collide
## meaningful if the user can guess the missed labels
X.Axis.CullingLabel=false;
X.Axis.AutoScale=true
## -- Determines end of an axis for a default axis.
## -- For log-scale its a power of ten.
X.Axis.AxisEnd=6.0
## -- Determines start of axis.
## -- For AUTO_SCALE, selection of axis start is automatic,
## -- for log scale its a power of ten.
X.Axis.AxisStart=0.0
## -- placed bar within axis, set true for bar charts
X.Axis.BarScaling=true
X.Axis.GridVis=false
X.Axis.LabelAngle=0
#### -- Axis label color
X.Axis.LabelColor.Red=0
X.Axis.LabelColor.Blue=0
X.Axis.LabelColor.Green=0

X.Axis.LabelFont.Name=Times New Roman
X.Axis.LabelFont.Style=plain
X.Axis.LabelFont.Size=12
#### -- Label Format determines how the label must be redendered,
#### -- eg. first three letters of month name, basically the defined method
#### -- is applied to the actual label
X.Axis.LabelFormat=null
## For double quantities such as currency
X.Axis.LabelPrecision=2
## Determines whether the label is visible
X.Axis.LabelVis=true
## Determines whether the axis line is visible
X.Axis.LineVis=true
## Determines whether the scale is log based
X.Axis.LogScaling=false
X.Axis.MajTickLength=5
X.Axis.MajTickVis=true
X.Axis.MinTickLength=2
X.Axis.MinTickVis=false

```

```

X.Axis.NumGrids=5
X.Axis.NumLabels=5
X.Axis.NumMajTicks=5
X.Axis.NumMinTicks=10

## reduce LlX to left shift axis position on canvas
## increase for right shift, that is along the X direction
X.Axis.Plotarea.LlX=0.2
#### -- reduce LlX to left shift on the canvas the axis start
#### -- increase LlX to right shift axis start on canvas
X.Axis.Plotarea.LlY=0.2
## similar to LlX and LlY, but controls upper right corner
X.Axis.Plotarea.UrX=0.8
X.Axis.Plotarea.UrY=0.8
## true sets the title perpendicular to the axis, in the middle
X.Axis.TitleRotated=true
X.Axis.TitleColor.Red=0
X.Axis.TitleColor.Blue=0
X.Axis.TitleColor.Green=0
X.Axis.TitleFont.Family=Times New Roman
X.Axis.TitleFont.Name=Times New Roman
X.Axis.TitleFont.Style=plain
X.Axis.TitleFont.Size=12

X.Axis.TitleString=Remember to set X axis title!
#### -- ignore this, not planning to retrieve objects using
#### -- mouse click events
X.Axis.UseDisplayList=false

## X.Axis.GridGc.FillColor.Red=0
## X.Axis.GridGc.FillColor.Blue=0
## X.Axis.GridGc.FillColor.Green=0
## X.Axis.GridGc.Image -- unimplemented
## X.Axis.GridGc.LineColor.Red=0
## X.Axis.GridGc.LineColor.Blue=0
## X.Axis.GridGc.LineColor.Green=0
## X.Axis.GridGc.LineWidth=1
## X.Axis.GridGc.MarkerColor.Red=0
## X.Axis.GridGc.MarkerColor.Blue=0
## X.Axis.GridGc.MarkerColor.Green=0

## valid values = Left, Right, Top, Bottom

```

```

## X.Axis.Side=Left

## X.Axis.TickGc.FillColor.Red=0
## X.Axis.TickGc.FillColor.Blue=0
## X.Axis.TickGc.FillColor.Green=0
## X.Axis.TickGc.Image -- unimplemented
## X.Axis.TickGc.LineColor.Red=0
## X.Axis.TickGc.LineColor.Blue=0
## X.Axis.TickGc.LineColor.Green=0
## X.Axis.TickGc.LineWidth=1
## X.Axis.TickGc.MarkerColor.Red=0
## X.Axis.TickGc.MarkerColor.Blue=0
## X.Axis.TickGc.MarkerColor.Green=0

## X.Axis.LineGc.FillColor.Red=0
## X.Axis.LineGc.FillColor.Blue=0
## X.Axis.LineGc.FillColor.Green=0
## X.Axis.LineGc.Image -- unimplemented
## X.Axis.LineGc.LineColor.Red=0
## X.Axis.LineGc.LineColor.Blue=0
## X.Axis.LineGc.LineColor.Green=0
## X.Axis.LineGc.LineWidth=1
## X.Axis.LineGc.MarkerColor.Red=0
## X.Axis.LineGc.MarkerColor.Blue=0
## X.Axis.LineGc.MarkerColor.Green=0

# -----
# Class name = javachart.chart.Axis
# Specific to Y axis
# -----

## to skip labels that collide
## meaningful if the user can guess the missed labels
Y.Axis.CullingLabel=false;
Y.Axis.AutoScale=true
## -- Determines end of an axis for a default axis.
## -- For log-scale its a power of ten.
Y.Axis.AxisEnd=6.0
## -- Determines start of axis.
## -- For AUTO_SCALE, selection of axis start is automatic,
## -- for log scale its a pwoer of ten.
Y.Axis.AxisStart=0.0

```

```

## -- placed bar within axis, set true for bar charts
Y.Axis.BarScaling=true
Y.Axis.GridVis=false
Y.Axis.LabelAngle=0
#### -- Axis label color
Y.Axis.LabelColor.Red=0
Y.Axis.LabelColor.Blue=0
Y.Axis.LabelColor.Green=0

Y.Axis.LabelFont.Name=Times New Roman
Y.Axis.LabelFont.Style=plain
Y.Axis.LabelFont.Size=12
#### -- Label Format determines how the label must be redenered,
#### -- eg. first three letters of month name, basically the defined method
#### -- is applied to the actual label
Y.Axis.LabelFormat=null
## For double quantities such as currency
Y.Axis.LabelPrecision=2
## Determines whether the label is visible
Y.Axis.LabelVis=true
## Determines whether the axis line is visible
Y.Axis.LineVis=true
## Determines whether the scale is log based
Y.Axis.LogScaling=false
Y.Axis.MajTickLength=5
Y.Axis.MajTickVis=true
Y.Axis.MinTickLength=2
Y.Axis.MinTickVis=false
Y.Axis.NumGrids=5
Y.Axis.NumLabels=5
Y.Axis.NumMajTicks=5
Y.Axis.NumMinTicks=10

## reduce LlX to left shift axis position on canvas
## increase for right shift, that is along the X direction
Y.Axis.Plotarea.LlX=0.2
#### -- reduce LlX to left shift on the canvas the axis start
#### -- increase LlX to right shift axis start on canvas
Y.Axis.Plotarea.LlY=0.2
## similar to LlX and LlY, but controls upper right corner
Y.Axis.Plotarea.UrX=0.8
Y.Axis.Plotarea.UrY=0.8

```

```

## true sets the title perpendicular to the axis, in the middle
X.Axis.TitleRotated=true
Y.Axis.TitleColor.Red=0
Y.Axis.TitleColor.Blue=0
Y.Axis.TitleColor.Green=0
Y.Axis.TitleFont.Family=Times New Roman
Y.Axis.TitleFont.Name=Times New Roman
Y.Axis.TitleFont.Style=plain
Y.Axis.TitleFont.Size=12

Y.Axis.TitleString=Remember to set X axis title!
#### -- ignore this, not planning to retrieve objects using
#### -- mouse click events
Y.Axis.UseDisplayList=false

## Y.Axis.GridGc.FillColor.Red=0
## Y.Axis.GridGc.FillColor.Blue=0
## Y.Axis.GridGc.FillColor.Green=0
## Y.Axis.GridGc.Image -- unimplemented
## Y.Axis.GridGc.LineColor.Red=0
## Y.Axis.GridGc.LineColor.Blue=0
## Y.Axis.GridGc.LineColor.Green=0
## Y.Axis.GridGc.LineWidth=1
## Y.Axis.GridGc.MarkerColor.Red=0
## Y.Axis.GridGc.MarkerColor.Blue=0
## Y.Axis.GridGc.MarkerColor.Green=0

## valid values = Left, Right, Top, Bottom
## Y.Axis.Side=Left

## Y.Axis.TickGc.FillColor.Red=0
## Y.Axis.TickGc.FillColor.Blue=0
## Y.Axis.TickGc.FillColor.Green=0
## Y.Axis.TickGc.Image -- unimplemented
## Y.Axis.TickGc.LineColor.Red=0
## Y.Axis.TickGc.LineColor.Blue=0
## Y.Axis.TickGc.LineColor.Green=0
## Y.Axis.TickGc.LineWidth=1
## Y.Axis.TickGc.MarkerColor.Red=0
## Y.Axis.TickGc.MarkerColor.Blue=0
## Y.Axis.TickGc.MarkerColor.Green=0

```

```

## Y.Axis.LineGc.FillColor.Red=0
## Y.Axis.LineGc.FillColor.Blue=0
## Y.Axis.LineGc.FillColor.Green=0
## Y.Axis.LineGc.Image -- unimplemented
## Y.Axis.LineGc.LineColor.Red=0
## Y.Axis.LineGc.LineColor.Blue=0
## Y.Axis.LineGc.LineColor.Green=0
## Y.Axis.LineGc.LineWidth=1
## Y.Axis.LineGc.MarkerColor.Red=0
## Y.Axis.LineGc.MarkerColor.Blue=0
## Y.Axis.LineGc.MarkerColor.Green=0

# -----
# Optional Properties:
# -----

## Set the Bar baseline.
##Bar.Baseline=0.0
## sets the cluster width
##Bar.ClusterWidth=0.8
## Set to true to clip bars at Plotarea boundaries.
## Bar.DoClip=false

# -----
# Optional Properties: Line
# -----
## true clips lines at the plot area boundary
## Line.Clip=false
# -----
# Optional Properties: Pie
# -----

Pie.LabelColor.Red=0
Pie.LabelColor.Blue=0
Pie.LabelColor.Green=0
Pie.LabelFont.Name=Times New Roman
Pie.LabelFont.Style=plain
Pie.LabelFont.Size=12

Pie.LabelPosition=2

```

```

Pie.PercentLabelsOn=true
Pie.StartDegrees=0
Pie.TextLabelsOn=false
Pie.ValueLabelsOn=false
#### -- if you want a circular pie, scale Height and Width to
#### -- be equal in terms of pixels after you've taken into
#### -- consideration true canvas height and width
Pie.Height=0.6
Pie.Width=0.6
#### -- this is the center of the pie, do you want it in the center
#### -- of the canvas or to one side. Elect side if the labels to be
#### -- rendered on the legend are long.
#### -- choose up or down, if you have more vertical real estate on the
#### -- html page
Pie.XLoc=0.5
Pie.YLoc=0.5

# -----
# General chart properties
# -----

Chart.LegendVisible=false
Chart.Name=MyChart
## x and y offset determine the three dimensional effect
Chart.ThreeD=false
Chart.XOffset=0
Chart.YOffset=0
Chart.YAxisVisible=true
Chart.XAxisVisible=true
## -- The chart quality has a default value of 1. It can take values
## -- from 0 to 1, where 0 is the poorest quality, while 1 is the best
## -- 0.75 is a good balance between image size and quality
Chart.Quality=1.0

# -----
# Plotarea Graphic Component properties
# -----

## Plotarea.Gc.FillColor.Red=0
## Plotarea.Gc.FillColor.Blue=0
## Plotarea.Gc.FillColor.Green=0
## Plotarea.Gc.Image -- unimplemented
## Plotarea.Gc.LineColor.Red=0
## Plotarea.Gc.LineColor.Blue=0

```

```

## Plotarea.Gc.LineColor.Green=0
## Plotarea.Gc.LineWidth=1
## Plotarea.Gc.MarkerColor.Red=0
## Plotarea.Gc.MarkerColor.Blue=0
## Plotarea.Gc.MarkerColor.Green=0

# -----
# Background properties
# -----

## Background.Gc.FillColor.Red=0
## Background.Gc.FillColor.Blue=0
## Background.Gc.FillColor.Green=0
## Background.Gc.Image -- unimplemented
## Background.Gc.LineColor.Red=0
## Background.Gc.LineColor.Blue=0
## Background.Gc.LineColor.Green=0
## Background.Gc.LineWidth=1
## Background.Gc.MarkerColor.Red=0
## Background.Gc.MarkerColor.Blue=0
## Background.Gc.MarkerColor.Green=0

## Background.SubTitleColor.Red=0
## Background.SubTitleColor.Blue=0
## Background.SubTitleColor.Green=0
## Background.SubTitleFont.name=Times New Roman
## Background.SubTitleFont.style=plain
## Background.SubTitleFont.size=12
## Background.SubTitleString=null

## Background.TitleColor.Red=0
## Background.TitleColor.Blue=0
## Background.TitleColor.Green=0
## Background.TitleFont.Name=Times New Roman
## Background.TitleFont.Style=plain
## Background.TitleFont.Size=12
## Background.TitleString=null

## Favorite.1.Color.Red=0
## Favorite.1.Color.Blue=0
## Favorite.1.Color.Green=0

## Favorite.2.Color.Red=0

```

```

## Favorite.2.Color.Blue=0
## Favorite.2.Color.Green=0
## Favorite.3.Color.Red=0
## Favorite.3.Color.Blue=0
## Favorite.3.Color.Green=0
## Favorite.4.Color.Red=0
## Favorite.4.Color.Blue=0
## Favorite.4.Color.Green=0

## Favorite.5.Color.Red=0
## Favorite.5.Color.Blue=0
## Favorite.5.Color.Green=0

```

13.9.2 NW_LocSummary.ALF

```

<?xml version="1.0"?>
<!DOCTYPE ALF [
    <!-- An element of type ALF must contain following subelements -->
    <![ELEMENT ALF (VERSION, DATA_GROUP, DDF, SWITCH, HOME, TEMPLATES,
CONTENTS, CONDITIONS, PROFILES, BUSINESSCONDITIONS, RECORDS, PAGE_ELEMENTS,
composition-specs)>
    <!-- An element of type VERSION contains a mixture of character data -->
    <![ELEMENT VERSION (#PCDATA)>
    <!-- An element of type DATA_GROUP contains a mixture of character data --
>
    <![ELEMENT DATA_GROUP (#PCDATA)>
    <!-- An element of type DDF contains a mixture of character data -->
    <![ELEMENT DDF (#PCDATA)>
    <!-- An element of type SWITCH consist of Optional Statement element -->
    <![ELEMENT SWITCH (Statement?)>
    <!-- An element of type Statement can contain three subelements. Firstly
it must
have Condition element and Action1 element. This is Optionally followed by the
Action2 element -->
    <![ELEMENT Statement (Condition, Action1, Action2?)>
    <![ATTLIST Statement
Profile (Y | N) #IMPLIED
>
    <!-- An element of type Condition contains a mixture of character data -->
    <![ELEMENT Condition (#PCDATA)>
    <!-- An element of type Action1 contains a mixture of character data -->
    <![ELEMENT Action1 (#PCDATA)>
    <!-- An element of type Action2 contains a mixture of character data or
Statement elements in any order-->
    <![ELEMENT Action2 (#PCDATA | Statement)*>

```

```

<!ELEMENT HOME (DefaultTemplate, Statement?)>
<!ELEMENT DefaultTemplate (#PCDATA)>
<!ELEMENT TEMPLATES (Template)+>
<!ELEMENT Template (SECTIONS, CHARTS, GROUPS, GroupTemplate*)>
<!ATTLIST Template
Name CDATA #REQUIRED
>

<!ELEMENT FormatSpecification (#PCDATA)>
<!ELEMENT Action (#PCDATA)>
<!ELEMENT SECTIONS (Section*)>
<!ELEMENT Section (Statement+ | (FormatSpecification, Action)+)>
<!ATTLIST Section
Name CDATA #REQUIRED
Promotional CDATA #IMPLIED
>

<!ELEMENT CHARTS (Chart*)>
<!ELEMENT Chart EMPTY>
<!ATTLIST Chart
Name CDATA #REQUIRED
RecordName CDATA #REQUIRED
TopTitle CDATA #REQUIRED
BottomTitle CDATA #REQUIRED
LeftTitle CDATA #REQUIRED
RightTitle CDATA #REQUIRED
XField CDATA #REQUIRED
YField CDATA #REQUIRED
Key CDATA #REQUIRED
StackedStyle CDATA #REQUIRED
ColorScheme CDATA #REQUIRED
GridLines CDATA #REQUIRED
Full3D CDATA #REQUIRED
AngleX CDATA #REQUIRED
AngleY CDATA #REQUIRED
Attribute CDATA #REQUIRED
MarkerVolume CDATA #REQUIRED
Shadow CDATA #REQUIRED
MultiShape CDATA #REQUIRED
Dimension_3D CDATA #REQUIRED
View3DDepth CDATA #REQUIRED
Type CDATA #REQUIRED
CGITimeSpan CDATA #REQUIRED
BackgroundColor CDATA #REQUIRED
ForegroundColor CDATA #REQUIRED

```

```

        Height CDATA #REQUIRED
        Width CDATA #REQUIRED
        LegendShow CDATA #REQUIRED
        LegendToolSize CDATA #REQUIRED
        LegendToolStyle CDATA #REQUIRED
        HidePieLegend CDATA #REQUIRED
        SeriesColor CDATA #REQUIRED
        LeftGap CDATA #REQUIRED
        RightGap CDATA #REQUIRED
        ImgQuality CDATA #REQUIRED
        ImgSmooth CDATA #REQUIRED
        AddValueToLegend CDATA #REQUIRED
        BaseURL CDATA #REQUIRED
        UNIXChart CDATA #REQUIRED
    >

    <!--ELEMENT GROUPS (Group*)-->
    <!--ELEMENT Group (Statement)-->
    <!--ATTLIST Group
    Name CDATA #REQUIRED
    >

    <!--ELEMENT GroupTemplate (SECTIONS, CHARTS, GROUPS, GroupTemplate*)-->
    <!--ATTLIST GroupTemplate
    Name CDATA #REQUIRED
    >

    <!--ELEMENT CONTENTS (Content)+-->
    <!--ELEMENT Content (#PCDATA)-->
    <!--ATTLIST Content
    Name CDATA #REQUIRED
    Type (MainTemplate | RGTemplate1 | RGTemplate2 | SectionTemplate |
ALF | Image | Text | Active | GlobalAction) "MainTemplate"
    Parent CDATA #REQUIRED
    ParentTemplate CDATA #REQUIRED
    >

    <!--ELEMENT CONDITIONS (SavedCondition)*-->
    <!--ELEMENT SavedCondition (#PCDATA)-->
    <!--ATTLIST SavedCondition
    Name CDATA #REQUIRED
    SavedConditionProfile (Y | N) #REQUIRED
    >

    <!--ELEMENT PROFILES (Profile)*-->
    <!--ELEMENT Profile (#PCDATA)-->
    <!--ATTLIST Profile
    Name CDATA #REQUIRED

```

```

>
<!ELEMENT BUSINESSCONDITIONS (BusinessCondition)*>
<!ELEMENT BusinessCondition (#PCDATA)>
<!--ATTLIST BusinessCondition
Name CDATA #REQUIRED
-->
>
<!ELEMENT RECORDS (Record)*>
<!ELEMENT Record (#PCDATA)>
<!--ATTLIST Record
Name CDATA #REQUIRED
ApplyAll (Y | N) #REQUIRED
PresentationTable CDATA #IMPLIED
-->
>
<!ELEMENT PAGE_ELEMENTS (PageElement)*>
<!ELEMENT PageElement (#PCDATA)>
<!--ATTLIST PageElement
Name CDATA #REQUIRED
Type (Table | Group) #REQUIRED
Enable (yes | no) #REQUIRED
Mode (line | occurrence) #REQUIRED
SetSize CDATA #REQUIRED
Occurrences CDATA #REQUIRED
-->
>
<!--ELEMENT composition-specs ((sort-spec | filter-spec | select-spec |
arithmetic-spec)*, combine-spec)>
<!--ELEMENT sort-spec (sorted-element, sort-by-element)+>
<!--ATTLIST sort-spec
name CDATA #REQUIRED
mode (Table | Group) #REQUIRED
-->
>
<!--ELEMENT sorted-element (#PCDATA)>
<!--ELEMENT sort-by-element (#PCDATA)>
<!--ATTLIST sort-by-element
data-type CDATA #REQUIRED
format-string CDATA #REQUIRED
direction (a | d) #REQUIRED
-->
>
<!--ELEMENT filter-spec (filtered-element, filtered-by-element, filter-
expression)+>
<!--ATTLIST filter-spec
name CDATA #REQUIRED
mode (Table | Group) #REQUIRED
-->
>
<!--ELEMENT filtered-element (#PCDATA)>

```

```

<!ELEMENT filtered-by-element (#PCDATA)>
<!ELEMENT filter-expression (#PCDATA)>
<!ELEMENT select-spec (selected-element, selected-by-element)+>
<!ATTLIST select-spec
  name CDATA #REQUIRED
  mode (Table | Group) #REQUIRED
>

<!ELEMENT selected-element (#PCDATA)>
<!ELEMENT selected-by-element (#PCDATA)>
<!ATTLIST selected-by-element
  data-type CDATA #REQUIRED
  format-string CDATA #REQUIRED
  direction (Top | Bottom) #REQUIRED
  default-count CDATA #REQUIRED
>

<!ELEMENT combine-spec (combine-element)*>
<!ELEMENT combine-element (#PCDATA)>
<!ELEMENT arithmetic-spec (arithmetic-element, arithmetic-by-element)>
<!ATTLIST arithmetic-spec
  name CDATA #REQUIRED
>

<!ELEMENT arithmetic-element (#PCDATA)>
<!ELEMENT arithmetic-by-element (#PCDATA)>
<!ATTLIST arithmetic-by-element
  data-type CDATA #REQUIRED
  format-string CDATA #REQUIRED
  mode (total | count) #REQUIRED
  output-format-string CDATA #REQUIRED
>
]>
<ALF>

<VERSION>3.0</VERSION>
<DATA_GROUP>Local_Summary</DATA_GROUP>
<DDF>C:/EDCSbd/samples/NatlWireless/NW_LocSummary.ddf</DDF>
<SWITCH/>
<HOME>
  <DefaultTemplate>Default_Template</DefaultTemplate>
</HOME>
<TEMPLATES>
  <Template Name="Default_Template">
    <SECTIONS/>
    <CHARTS>

```

```

<Chart Name="LocalLineSummary_0"
RecordName="LocalLineSummary" TopTitle="Top Lable" BottomTitle="Bottom Lable"
LeftTitle="" RightTitle="" XField="LocalLinePhNo" YField="LocalLineAmt" Key="0"
StackedStyle="0" ColorScheme="0" GridLines="0" Full3D="0" AngleX="0" AngleY="0"
Attribute="0" MarkerVolume="0" Shadow="0" MultiShape="0" Dimension_3D="0"
View3DDepth="0" Type="1" CGITimeSpan="" BackgroundColor="White"
ForegroundColor="Black" Height="300" Width="400" LegendShow="1"
LegendToolSize="100" LegendToolStyle="167116800" HidePieLegend="0" SeriesColor=""
LeftGap="40" RightGap="40" ImgQuality="75" ImgSmooth="0" AddValueToLegend="0"
BaseURL="/Chart" UNIXChart="Pie"/>

<Chart Name="LocalChargeSummary_1"
RecordName="LocalChargeSummary" TopTitle="Top Lable" BottomTitle="Bottom Lable"
LeftTitle="" RightTitle="" XField="LocalChargeAmt" YField="LocalChargeDesc"
Key="1" StackedStyle="0" ColorScheme="0" GridLines="0" Full3D="0" AngleX="0"
AngleY="0" Attribute="0" MarkerVolume="0" Shadow="0" MultiShape="0"
Dimension_3D="0" View3DDepth="0" Type="1" CGITimeSpan="" BackgroundColor="White"
ForegroundColor="Black" Height="300" Width="400" LegendShow="1"
LegendToolSize="100" LegendToolStyle="167116800" HidePieLegend="0" SeriesColor=""
LeftGap="40" RightGap="40" ImgQuality="75" ImgSmooth="0" AddValueToLegend="0"
BaseURL="/eaSample" UNIXChart="Pie"/>

</CHARTS>

<GROUPS/>

</Template>

</TEMPLATES>

<CONTENTS>

<Content Name="Default_Template" Type="MainTemplate" Parent=""
ParentTemplate=""><![CDATA[C:/EDCSbd/samples/NatlWireless/NW_LocSummary.htm]]></C
ontent>

</CONTENTS>

<CONDITIONS/>

<PROFILES/>

<BUSINESSCONDITIONS/>

<RECORDS>

<Record Name="CustAddress" ApplyAll="Y"><![CDATA[<table border=1
width="100%">

<TBODY>

<tr>

<td height=% width=%><font color=#5c00d9 face=Arial

size=2><STRONG>[E] CustAddressLine [/E] </STRONG></font></td></tr></TBODY></table>]]
></Record>

<Record Name="LocalChargeSummary" ApplyAll="Y"><![CDATA[<TABLE
border=1 width="100%">

<TBODY>

<TR>

<TD height=% width=%><FONT color=#000000 face=Arial

size=2>[E] LocalChargeDesc [/E] </FONT></TD>

<TD align=right height=% width=%><FONT color=#000000 face=Arial

size=2>[E] LocalChargeAmt [/E] </FONT></TD></TR></TBODY></TABLE>]]></Record>

<Record Name="LocalLineSummary" ApplyAll="Y"><![CDATA[<table
border=1 width="100%">

<TBODY>

<tr>

```

```

<td height=% width=%><font color=#000000 face=Arial
size=2>[E]LocalLinePhNo[/E]</font></td>
<td align=right height=% width=%><font color=#000000 face=Arial
size=2>[E]LocalLineAmt[/E]</font></td></tr></TBODY></table>]]></Record>
</RECORDS>
<PAGE_ELEMENTS/>
<composition-specs>
    <combine-spec/>
</composition-specs>
</ALF>

```