Oracle

Primavera P6 EPPM Web and Pro API Programming Guide for On-Premises

Version 23

February 2024

Contents

About This Guide

Scope

This guide describes how to do the following using the P6 Integration API and P6 Professional Integration API applications:

- Access data
- Use business objects

Audience

System administrators should use this guide.

About This Guide

This guide assumes that you have already installed a supported version of JDK, and have familiarity with XML. For more information, see the *Tested Configurations* document.

In This Section

About Personal Information

Personal information (PI) is any piece of data which can be used on its own or with other information to identify, contact or locate an individual or identify an individual in context. This information is not limited to a person's name, address, and contact details, for example a person's IP address, phone IMEI number, gender, and location at a particular time could all be personal information. Organizations are responsible for ensuring the privacy of PI wherever it is stored, including in back-ups, locally stored downloads, and data stored in development environments.

Caution: Personal information (PI) may be at risk of exposure. Depending on local data protection laws organizations may be responsible for mitigating any risk of exposure.

Introduction

The P6 Integration API and P6 Professional Integration API are both flexible, object-oriented, cross platform, and Java-based interfaces. The P6 Integration API is used only in a P6 EPPM environment.

The P6 Integration API is designed to run in one of two modes: Local or Remote. The P6 Professional Integration API is designed to run in one mode: Local. For more information about Local and Remote modes, see ***Modes of Operation***. It is important to understand that both modes are available only if you are using the P6 Integration API, which can be done only in a P6 EPPM environment.

This document provides information on how to use these APIs. See the P6   for information on installing and configuring the API and system requirements. For information on how to use specific classes, see the associated Javadoc.

Note: The P6 Integration API is supported in this release, but might not be supported in a future release. Oracle recommends using P6 EPPM Web Services for integrations.

What's New in P6 Integration API

New Classes and Enumerations

The following classes and fields have been added to P6 Integration API.

New Classes

- LeanTask
- LeanTaskHelper
- Status
- StatusCompletion
- StatusDates
- TaskStatusCompletion
- TaskStatusDates
- UserConsent
- UserConsentHelper

New Methods

- New Activity Methods
  - getOwnerIDArray()
  - getOwnerNamesArray()
  - getPerformancePercentCompleteByLaborUnits()
  - setOwnerIDArray(String[])
  - setOwnerNamesArray(String[])
  - setTaskStatusCompletion(TaskStatusCompletion)
  - setTaskStatusDates(TaskStatusDates)

  - setTaskStatusIndicator(boolean)
  - createLeanTask(LeanTask)
  - createLeanTasks(LeanTask[])
  - deleteLeanTasks(LeanTask[])
- New ActivityFilter Methods
  - getFilterCriteriaConfig()
  - getFilterType()
  - setActivityFilterId(ObjectId)
  - setActivityFilterName(String)
  - setFilterCriteriaConfig(String)
  - setFilterType(String)
  - setUserId(ObjectId)
- New BaselineProject Methods
  - getResourcesCanAssignThemselvesToActivitiesOutsideOBSAccess()
  - setResourcesCanAssignThemselvesToActivitiesOutsideOBSAccess(boolean)
  - loadAllLeanTasks(String[], String, String)
- New BusinessObject Methods
  - setEarlyDate(int, Date)
- New EnterpriseLoadManager Methods
  - loadUserConsent(String[], String, String)
- New GlobalPreferences Methods
  - getEPPMConsentMessage()
  - getEPPMEnableConsent()
  - getResourcesCanAssignThemselvesToActivitiesOutsideTheirOBSAccess()
  - setResourcesCanAssignThemselvesToActivitiesOutsideTheirOBSAccess(boolean)
  - setTeamMemberConsentMessage(String)
  - setTeamMemberEnableConsent(String)
- New Project Methods
  - getPerformancePercentCompleteByLaborUnits()
  - getResourcesCanAssignThemselvesToActivitiesOutsideOBSAccess()
  - setResourcesCanAssignThemselvesToActivitiesOutsideOBSAccess(boolean)
  - loadAllLeanTasks(String[], String, String)
- New ResourceAssignment Methods
  - getPlannedCurve()
  - getRemainingCurve()
  - setPlannedCurve(String)
  - setRemainingCurve(String)
  - loadWithLiveSpreadActivityOrAssignment(Session, String[], ObjectId[], String[], SpreadPeriodType, Date, Date, boolean)
- New UpdateBaselineOption Methods

- getNewActivityInformation()
- getNewBudgetUnitsCost()
- setNewActivityInformation(boolean)
- setNewBudgetUnitsCost(boolean)

New Fields

- New Activity Fields
  - OwnerIDArray
  - OwnerNamesArray
  - PerformancePercentComplete
  - TaskStatusCompletion
- New ActivityFilter Fields
  - FilterCriteriaConfig
  - FilterType
  - UserId
- New UDFType Fields
  - Formula
  - SummaryMethod
- New UpdateBaselineOption Fields
  - NewActivityInformation
  - NewBudgetUnitsCost
- New BaselineProject Fields
  - ResourcesCanAssignThemselvesToActivitiesOutsideOBSAccess
- New GlobalPreferences Fields
  - EPPMConsentMessage
  - EPPMEnableConsent
  - ResourcesCanAssignThemselvesToActivitiesOutsideTheirOBSAccess
  - TeamMemberConsentMessage
  - TeamMemberEnableConsent
- New IssueHistory Fields
  - ProjectIssueObjectId
  - ProjectObjectId
- New Project Fields
  - PerformancePercentCompleteByLaborUnits
  - ResourcesCanAssignThemselvesToActivitiesOutsideOBSAccess

Modes of Operation

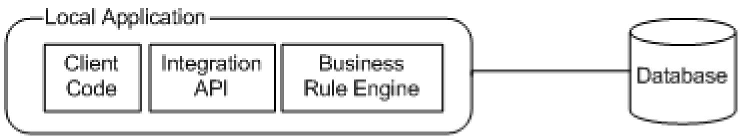
As discussed in the "Introduction" topic, the P6 Integration API is designed to run in Local Mode or Remote, while the P6 Professional API can run in Local Mode only.

For more information about Local and Remote Modes, proceed to the applicable subsection below, depending on the API you use:

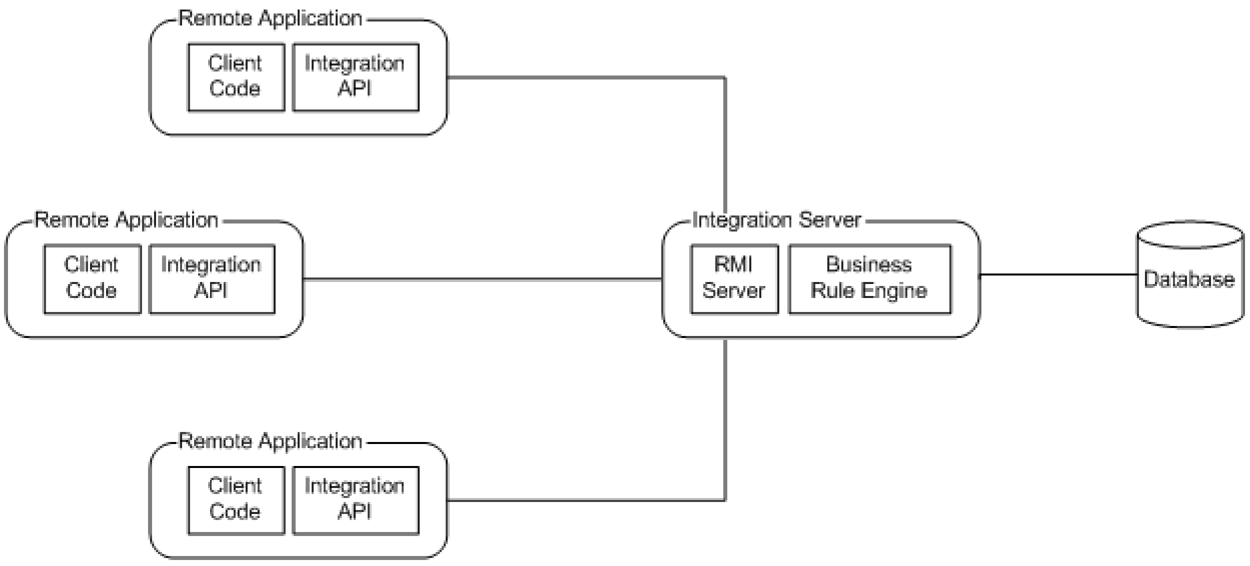
- Modes of Operation If Using the P6 Integration API
- Modes of Operation If Using the P6 Professional API

Modes of Operation If Using the P6 Integration API

In Local Mode, the client code runs in the same Java Virtual Machine (JVM) as the Integration server. Java Remote Method Invocation (RMI) is not used, and the Integration API communicates directly with the business rule code in the server (the Business Rule Engine). Local Mode is useful for when the API client code will be deployed on the same physical machine as the internal Business Rule Engine. It can also be useful for applications that require the improved performance achieved by avoiding the RMI layer.



In Remote Mode, the client code runs on a different machine than the Integration server and Java Remote Method Invocation (RMI) is used for communication. Multiple clients can communicate with the Integration server simultaneously.



Note: The maximum number of clients that can access a remote server at one time is approximately 50. This number can be less, depending on multiple factors including system hardware and network configuration.

There are three possible service modes for the RMI server: Standard, Compression, and SSL. By default, all three modes are enabled. The RMI server also requires the RMI Registry, which listens to port 9099 by default. You can change the default settings for the RMI server via the Administrator tool, which can be launched using admin.cmd (admin.sh for AIX, HPUX, Linux). The following settings can be found under Configurations\Custom\<Configuration Name>\Integration API Server\RMI:

Enable - Enables (true) or disables (false) the RMI server (default setting is true).

RegistryPort - Port for the RMI Registry (default setting is 9099).

StandardServiceEnable - Enables (true) or disables (false) the Standard service mode (default setting is true).

StandardServicePort - Port to use for Standard service mode. A setting of 0 (default) means that any available port will be used. If the server will be accessed across a firewall, you must set this to a specific port.

CompressionServiceEnable - Enables (true) or disables (false) the Compression service mode (default setting is true).

CompressionServicePort - Port to use for Compression service mode. A setting of 0 (default) means that any available port will be used. If the server will be accessed across a firewall, you must set this to a specific port.

SSLServiceEnable - Enables (true) or disables (false) the SSL service mode (default setting is true).

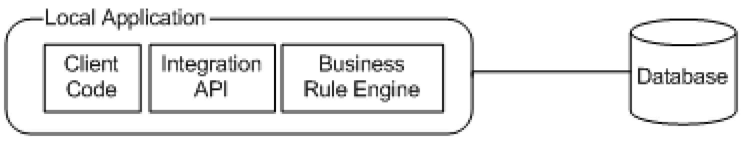
SSLServicePort - Port to use for SSL service mode. A setting of 0 (default) means that any available port will be used. If the server will be accessed across a firewall, you must set this to a specific port.

If the API is configured to use Remote Mode, the service mode can be chosen at runtime using the RMIURL helper class: standard, compression, and SSL modes are available.

Note: P6 EPPM Web Services should be considered as an alternative to using the Remote Mode of the API.

Modes of Operation If Using the P6 Professional API

If you are using the P6 Professional Integration API, then you can use the API in Local Mode only. Java Remote Method Invocation (RMI) is not used, and the Integration API communicates directly with the business rule code in the server (the Business Rule Engine), as illustrated in the following graphic:



Security

Application layer security for the APIs is similar to what is used by the Primavera client/server products. Global and project security profiles apply when using the APIs, so if a user attempts to perform an action that is restricted by a security profile, an exception will be thrown.

Network layer security is achievable by using SSL.

Note: If you are using the P6 Integration API, see the section entitled "Java Security Manager" in the *P6 EPPM Integration API Configuration Guide* for information on how to enable additional security through the use of a custom security policy.

Packages

Within the jar file, classes in the following packages can be accessed directly by client code:

com.primavera (base classes for Primavera exceptions)
com.primavera.common.exceptions (common exception classes)
com.primavera.common.value (value object classes)
com.primavera.common.value.spread (spread value classes)
com.primavera.integration.client (main classes, including Session, JobManager EnterpriseLoadManager, and GlobalObjectManager)
com.primavera.integration.client.bo (business object base class and iterator classes)
com.primavera.integration.client.bo.enm (typesafe enumerations)
com.primavera.integration.client.bo.helper (business object helper classes)
com.primavera.integration.client.bo.object (client business object classes)
com.primavera.integration.client.xml.xmlexporter
com.primavera.integration.client.xml.xmlimporter
com.primavera.integration.common (general common classes)
com.primavera.integration.network (exception classes for Remote Mode)
com.primavera.integration.util (utility and helper classes)

Other packages in the jar file contain code for internal use only.

Where to Begin

If you will be writing code against either API, the Java Software Development Kit (SDK, also known as the JDK) must be installed. The Integrated Development Environment (IDE) used to write code must work with the supported version of the JDK. If you will not be writing code, only the Java Runtime Environment (JRE) is required to be able to run applications written for the APIs.

Note: For information on what version of the Java Software Development Kit or Java Runtime Environment to use, see the *Tested Configurations* document.

When Using the P6 Integration API

The API client code for Remote Mode is contained in intgclient.jar. For Local Mode, the API code is contained in intgserver.jar. The jar files are installed in the lib directory under the Integration API installation directory. To successfully compile and run the code written against the API, you will need to include the appropriate jar file in your classpath.

For applications running in Local Mode, your classpath must include the other jar files that are installed in the lib directory under the Integration API installation directory. Local Mode applications must also have the System property "primavera.bootstrap.home" set to the location of the installation directory. This setting is used by the server to find the BREBootStrap.xml file.

When Using the P6 Professional API

The API client code is contained in intgserver.jar. This and other related jar files are installed in the lib directory under the Integration API installation directory. To successfully compile and run the code written against the API, you must include these jar files in your classpath.

Additionally, the System property "primavera.bootstrap.home" must also be set to the location of the installation directory. This setting is used by the server to find the BREBootStrap.xml file.

In This Section

Accessing Data in the API

To access data in the API, you must first establish a valid session.

Then proceed as follows:

- If you are using the P6 Professional API, see the next topic, "Example: Establishing a Session in Local Mode" for an example of how to establish a session and load a collection of projects. Remember the P6 Professional API can run in Local Mode only.

- If you are using the P6 Integration API, see the next two topics as applicable: "Example: Establishing a Session in Local Mode" and "Example: Establishing a Session in Remote Mode." (Code written for Local Mode is the same as code written for Remote Mode, except for calls to Session.getDatabaseInstances() and Session.login(), which require the appropriate information to be specified for finding the server.)

Example: Establishing a Session in Local Mode

The following code provides an example of how to establish a session in Local Mode and load a collection of projects:

```
import com.primavera.integration.client.Session;
import com.primavera.integration.client.EnterpriseLoadManager;
import com.primavera.integration.client.RMIURL;
import com.primavera.integration.common.DatabaseInstance;
import com.primavera.integration.client.bo.BOIterator;
import com.primavera.integration.client.bo.object.Project;
public class APITest
{

public static void main( String[] args )
{
Session session = null;
try
{
DatabaseInstance[] dbInstances = Session.getDatabaseInstances( RMIURL.getRmiUrl(
RMIURL.LOCAL_SERVICE ) );
// Assume only one database instance for now, and hardcode the username and
// password for this sample code
session = Session.login( RMIURL.getRmiUrl( RMIURL.LOCAL_SERVICE ),
dbInstances[0].getDatabaseId(), "admin", "admin" );
EnterpriseLoadManager elm = session.getEnterpriseLoadManager();
BOIterator<Project> boi = elm.loadProjects( new String[]{ "Name" }, null, "Name asc" );
while ( boi.hasNext() )
{
 Project proj = boi.next();
 System.out.println( proj.getName() );
}
}
catch ( Exception e )
{
// Best practices would involve catching specific exceptions. To keep this
// sample code short, we catch Exception
e.printStackTrace();
}
finally
{
if ( session != null ) session.logout(); }
}

}
```

Example: Establishing a Session in Remote Mode

The following code provides an example of how to establish a session in Remote Mode, using the standard service mode, and load a collection of projects.

Note: Remote Mode applies only if you are using the P6 Integration API.

Example 2: (see the Javadoc for RMIURL for information on how to specify other service modes):

```
import com.primavera.integration.client.Session;
import com.primavera.integration.client.EnterpriseLoadManager;
import com.primavera.integration.client.RMIURL;
import com.primavera.integration.common.DatabaseInstance;
import com.primavera.integration.client.bo.BOIterator;
import com.primavera.integration.client.bo.object.Project;
public class APITest { public static void main( String[] args )
{

Session session = null;
try
{
DatabaseInstance[] dbInstances = Session.getDatabaseInstances( RMIURL.getRmiUrl(
RMIURL.STANDARD_RMI_SERVICE, "localhost", 9099 ) );
// Assume only one database instance for now, and hardcode the username and
// password for this sample code. Assume the server is local for this sample code.
session = Session.login( RMIURL.getRmiUrl( RMIURL.STANDARD_RMI_SERVICE, "localhost", 9099 ),
dbInstances[0].getDatabaseId(), "admin", "admin" );
EnterpriseLoadManager elm = session.getEnterpriseLoadManager();
BOIterator<Project> boi = elm.loadProjects( new String[]{ "Name" }, null, "Name asc" );
while ( boi.hasNext() )
{
Project proj = boi.next();
System.out.println( proj.getName() );
}
}
catch ( Exception e )
{
// Best practices would involve catching specific exceptions. To keep this
// sample code short, we catch Exception e.printStackTrace();
}
finally
{
if ( session != null ) session.logout();
}

}
```

Core Functionality

In This Section

Session

Session is the main class used for communicating with the server. To establish a valid session, a static login method is used. The session reference can then be used to access other main objects, such as the EnterpriseLoadManager.

To log in, a valid database instance must be specified if multiple database instances are defined in the current configuration. Use the Administrator tool to see the settings of your configuration. If multiple configurations are defined in the database, check the "name" attribute of the Bootstrap\Configurations\BRE element in the BREBootStrap.xml file to determine the configuration used by your server installation.

Before logging in, you can retrieve a list of available database instances by calling Session.getDatabaseInstances().

Note: The only difference in client code for Local Mode and Remote Mode is the call to Session.getDatabaseInstances() and Session.login(). For Remote Mode, the code must specify the URL of the RMI server. You can use com.primavera.integration.client.RMIURL to generate the RMI URL for different remote modes: Standard, Compression, or SSL.

Session is not a singleton, which means you can establish multiple simultaneous communication sessions with various servers and/or database instances. This can be useful for integrating with multiple Oracle Primavera databases.

GlobalObjectManager

Retrieve the GlobalObjectManager instance for a particular session by calling Session.getGlobalObjectManager(). This object is used for accessing all global business objects: EPS, Projects, Resources, Roles, etc. In general, a business object is global if it is not a child of a different type of object.

From the GlobalObjectManager, global objects can be created, loaded, updated, and deleted. Each of these methods will cause the database to be accessed. When running in Remote Mode, each of these methods results in a remote call to the server, which in turn might update the database.

EnterpriseLoadManager

Retrieve the EnterpriseLoadManager instance for a particular session by calling Session.getEnterpriseLoadManager(). This object is used for accessing all business objects directly without having to follow a navigation model.

Business Objects

A business object is an encapsulation of business data and functionality that usually corresponds to a record in a particular database table. Business objects contain fields, exposed as properties. Get() methods exist for all fields, and set() methods exist only for writable fields. Most business objects contain an ObjectId field, which serves as the primary key for that object.

Note: Client-side business objects are transient and should not be reused. For example, when creating a new instance of a client-side business object, after you call the create() method to create the object in the database, the object should be reloaded from the database if you intend to use it. This will help ensure that the data is valid, based on the server-side business rules. This warning also applies to updating business objects; after calling update(), reload the object if you intend to use it further.

Load methods that cause multiple business objects to be loaded will return a BOIterator (a business object iterator), that can be used to iterate through the returned business objects. Similar to Java's java.util.Iterator class, it has both hasNext() and next() methods. Not all business objects are retrieved from the server at one time. As you iterate through the result set, more business objects are automatically loaded from the server as needed.

When loading an object, the fields to be loaded can be specified. If this parameter is null, the minimal fields necessary will be loaded. You can obtain lists of available fields by calling the following methods:

getAllFields() - Returns an array of all fields for this business object. Code assignment and UDF value field names are not included in this array. For more information, see the Special Handling of Codes and UDFs section below.

getRequiredCreateFields() - Returns an array of fields required to create this business object. Some business objects have fields listed in this array that are OR'ed. These fields will appear in the array separated by the "|" character. For example, the required create fields for Activity are "Id", and "ProjectObjectId|WBSObjectId," meaning the Id field must always be set, and either the ProjectObjectId or the WBSObjectId must be set (setting only the ProjectObjectId will cause the Activity to be created at the project-level).

getSpreadFields() - Returns an array of spread fields (unit and cost) for business objects that support spreads: EPS, Project, WBS, Activity, Role, Resource, and ResourceAssignment.

getMainFields() - Returns all fields supported by the business object, except for summary, code assignment, and UDF fields.

Note: In order to have the API perform optimally, only specify to load the fields that are absolutely needed.

Business objects can be loaded directly using static load() methods of the class itself, from a parent object, from the GlobalObjectManager if the object is global, or from the EnterpriseLoadManager. To run the API using the EnterpriseLoadManager, ensure that both the NLS_COMP and NLS_SORT parameters are set to the same value. Objects can be loaded by specifying an array of ObjectIds or by specifying a "where" clause and/or an "order by" clause. The where clause is used to filter the business objects when loaded.

- Where clauses that use Date data types must use the WhereClauseHelper to format the date value. See Load Activities Example below.
- Complex where clauses can be created using AND and OR.
- Where clauses follow SQL-92 grammar and support function calls of the SQL language, with some exceptions. Join statements and nested select statements are not supported. The DATEADD function of SQL Server is also not supported.

The following code examples demonstrate how to specify a where clause when loading business objects:

Example 1: Load all the projects that have an Id beginning with "API-Project," ordering by Id in ascending order:

```
EnterpriseLoadManager elm = session.getEnterpriseLoadManager();
BOIterator<Project> boi = elm.loadProjects( new String[]{ "Id", "Status",
"StartDate", "FinishDate" },
 "Id like 'API-Project%'", "Id asc" );
while ( boi.hasNext() )
{

Project proj = boi.next();
// Add code here to process each Project...

}
```

Example 2: Load activities from a project where the actual start is within a particular date range, ordering by Name in descending order:

```
SimpleDateFormat formatter = new SimpleDateFormat( "MM/dd/yyyy" );
Date date = formatter.parse( "03/03/2005" );
String dateBegin = WhereClauseHelper.formatDate( session, date );
date = formatter.parse( "03/09/2005" );
String dateEnd = WhereClauseHelper.formatDate( session, date );
String whereClause = "ActualStartDate between " + dateBegin + " and " + dateEnd;
BOIterator<Activity> boi = project.loadAllActivities( new String[]{ "Id", "Name"
}, whereClause, "Name desc" );
while ( boi.hasNext() )
{
```

Activity act = boi.next();
// Add code here to process each Activity...

}

Example 3: Load all active timesheets from a timesheet period:

```
BOIterator<Timesheet> boi = timesheetPeriod.loadTimesheets( new String[]{
"ResourceName", "ResourceId", "Status" }, "Status='" +
com.primavera.integration.client.bo.enm.TimesheetStatus.ACTIVE.getValue() +
"'", "" );



while ( boi.hasNext() )
{
```

Timesheet ts = boi.next();
// Add code here to process each Timesheet...

}

Additional Functionality

In This Section

JobManager

- If you are using the P6 Integration API, the Job Manager is used to invoke all asynchronous jobs: schedule, level, summarize, apply actuals, recalculateResourceAssignmentcosts, and store period performance. It is retrieved for a particular session by calling Session.getJobManager(). You can check the status of a particular asynchronous job by calling getJobStatus and passing in the job ID returned when the job was created. Other methods exist for deleting a job and getting a list of all jobs.
- If you are using the P6 Professional API, use the P6 Professional Job Service to invoke asynchronous jobs instead of the Job Manager. (Job Manager is used to invoke asynchronous jobs only when using the P6 Integration API.)

Batch Exception Handling

BatchException allows you to catch and collect all business rule exceptions without stopping the batch create/update transaction. After the whole batch create/update is processed, the data transaction is still rolled back but BatchException provides you a list of business rule exceptions that occurred during the process. Looping through the exception list, you can identify the problematic business objects. You might want to remove those from the transaction list and rerun the batch update/create again. Or you might want to fix the issue and rerun the batch update/create again.

Example: Catching BatchExceptions and looping through the exception list:

```
try
{

// call update or create here... ...

}
catch ( BatchException e )
{
```

```
// Display stack trace of batch exception
e.printStackTrace();
System.out.println();
// Display index and exception message of all exceptions in the batch exception
List exceptions = e.getExceptionList();
for ( int i = 0; i < exceptions.size(); i++ )
{
ServerException se = (ServerException)exceptions.get(i);
System.out.println( se.getSource() + " - " + se.getMessage() );
}

}
```

TimeStamps

Each business object now provides information about the user (getCreateUser()) and the date (getCreateDate()) when the business object was created. Similarly, the getLastUpdateUser() and getLastUpdateDate() return the user who updated the business object and the date it was updated respectively. The getCreateUser() and getLastUpdateUser() only return the name of the user. If the User object is needed, it will need to be loaded separately.

Example: Load all users that have been updated after June 30, 2005 at 6:00 AM. The results are ordered by the update date:

```
java.util.Calendar calendar = new java.util.GregorianCalendar();
// 2005-6-30 06:00AM
calendar.set( 2005, 5, 30, 6, 0, 0);
java.util.Date testDate = new java.util.Date( calendar.getTimeInMillis() );
String wc = WhereClauseHelper.formatDate( session, testDate );
BOIterator<User> boi = elm.loadUsers( User.getAllFields(), "LastUpdateDate > " +
wc, "LastUpdateDate asc" );
while ( boi.hasNext() )
{

User user = boi.next();
// Process user here...

}
```

Resource Security

Resource security allows you to restrict a user's ability to access resources. Restricted resource access means that a user has access to a parts of the resource hierarchy only. Privileges that control the resource hierarchy (add/edit/delete resource) still apply but only to those resources that the user has access to.

Resource access types

- Access to all resources

  If the User.AllResourceAccessFlag is True, the user has access to all resources and resource security does not apply.
- Restricted access to resources

  If the User.AllResourceAccessFlag is False, the user has limited access to resources.

  If the user is assigned to resources in the hierarchy, that user has access to the assigned resource and all their children. You can assign up to five resources to a user.
- If the user is not assigned to any resource in the hierarchy, that user does not have access to any resources.

Note: Admin Superusers always have access to all resources.

Accessing resources assigned to a project

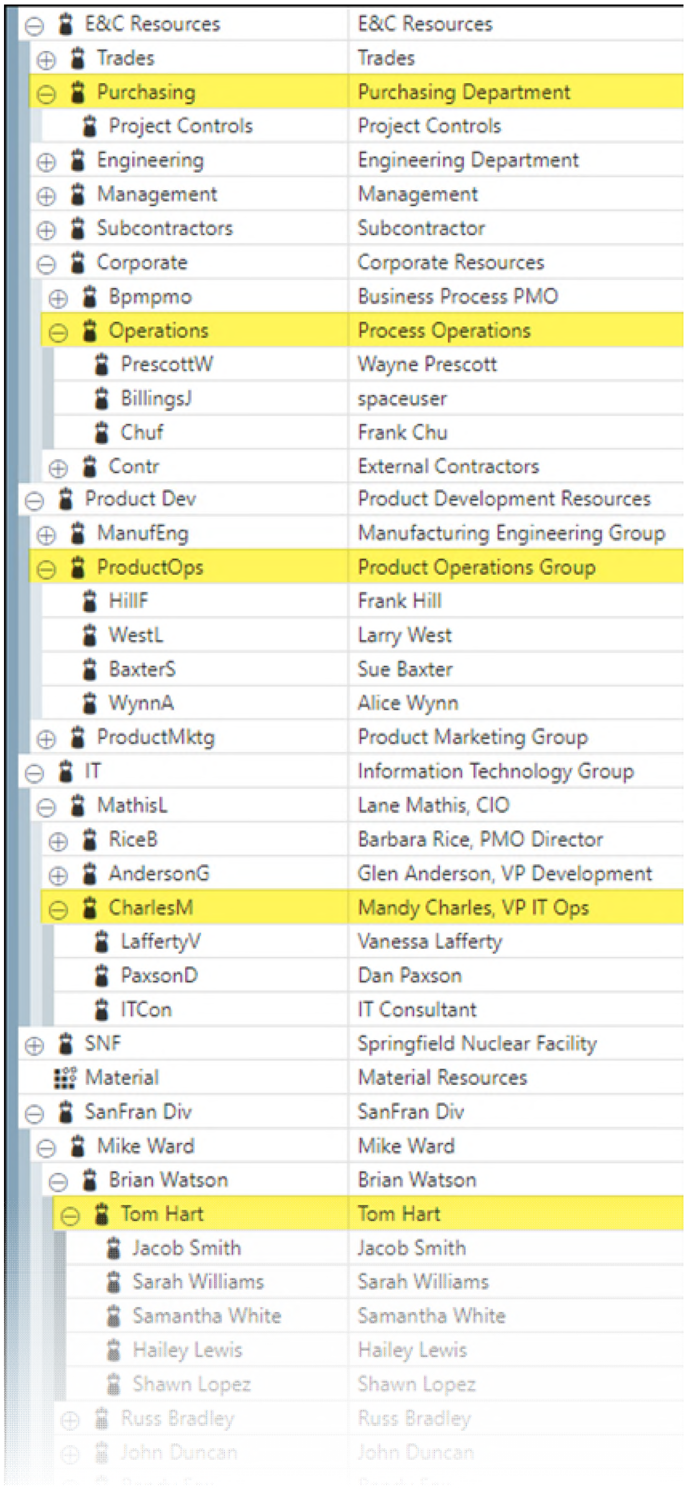
If a user can access a project, that user is able to see all resources assigned to that project (activity, issue, risk, WBS) even if they are outside the user's resource access nodes. The user can then reassign these resources anywhere, but will only be able to edit those that are under the user's resource access nodes. For more information on the resource security feature, refer to the P6 help (if using the P6 Integration API) or the P6 Professional help (if using the P6 Professional API).

The API implementation of resource security

Use the ResourceAccess business object to implement and maintain resource access in the API.

When importing resources using the API as a user with restricted resource access, import the resources to the first resource node that the user has access to, and into the highest resource node that the user has access to.

For example, in the structure below, if the user has access to the nodes Purchasing, Operations, ProductOps, CharlesM and Tom Hart, the user also has access to all the child resources of those nodes. When importing resources, if the user does not specify which branch to import the users into, import the resources under the Purchasing node.



E&C Resources	E&C Resources
Trades	Trades
Purchasing	Purchasing Department
Project Controls	Project Controls
Engineering	Engineering Department
Management	Management
Subcontractors	Subcontractor
Corporate	Corporate Resources
Bpmpmo	Business Process PMO
Operations	Process Operations
PrescottW	Wayne Prescott
BillingsJ	spaceuser
Chuf	Frank Chu
Contr	External Contractors
Product Dev	Product Development Resources
ManufEng	Manufacturing Engineering Group
ProductOps	Product Operations Group
HillF	Frank Hill
WestL	Larry West
BaxterS	Sue Baxter
WynnA	Alice Wynn
ProductMktg	Product Marketing Group
IT	Information Technology Group
MathisL	Lane Mathis, CIO
RiceB	Barbara Rice, PMO Director
AndersonG	Glen Anderson, VP Development
CharlesM	Mandy Charles, VP IT Ops
LaffertyV	Vanessa Lafferty
PaxsonD	Dan Paxson
ITCon	IT Consultant
SNF	Springfield Nuclear Facility
Material	Material Resources
SanFran Div	SanFran Div
Mike Ward	Mike Ward
Brian Watson	Brian Watson
Tom Hart	Tom Hart
Jacob Smith	Jacob Smith
Sarah Williams	Sarah Williams
Samantha White	Samantha White
Hailey Lewis	Hailey Lewis
Shawn Lopez	Shawn Lopez
Russ Bradley	Russ Bradley
John Duncan	John Duncan

For details on specific methods, refer to the JavaDoc.

Note: This note applies if you are using the P6 Integration API: On the P6 Users page, users are filtered based on your resource access settings. As an exception, Admin Superusers will always see all users. In the API, all users are loaded but the ability to modify a user's resource access settings is determined by your resource access settings. If the user is associated to a resource that is outside your resource access, you cannot change that user's resource access settings.

XML Exporter and Importer

In This Section

XMLExporter

The XML Exporters are used for writing business objects to XML. Every business object can be exported, either by specifying an array of ObjectIds or by specifying a where clause to use for loading the objects. Exporting non-global objects requires the parent object to be specified for the methods that have where clause parameters.

Another XMLExporter method, exportFullProject, exports a project and all of its child objects (such as WBS, Activities, ResourceAssignments, etc.). XML files created using exportFullProject can be imported using the XMLImporter.

When objects are exported, the fields to be exported can be specified. If the fields parameter is null, the default XML export fields will be used for each object. Since the default fields are the writable fields, you can obtain this list of default fields by calling getWritableFields() on each business object class.

To specify specific business object types or fields to be included in the export, use XMLExporterListener. See the exporter demo application (ExportDemoApp.java) for example code.

Note: All business objects can be written individually to XML even without using the XMLExporter. Simply call toString() on a business object instance and all fields currently loaded in that business object will be output to XML using the p6apibo.xsd schema.

XMLImporter

The current version of the XML Importer only supports importing projects and project-related data generated using XMLExporter.exportFullProject(). For the list of business objects and related data exported by the XMLExporter.exportFullProject(), refer to the XMLExporter JavaDoc.

The XMLImporter can also import XML generated data from other sources if the XML conforms to the required schema and all necessary data are provided. However, anything that is not in the full-project export, does not get imported, even if it is in the schema (for example, Users).

Note: Read-only fields or business objects do not get imported either, and this version of the XMLImporter does not support importing

Documents and related business objects.

An example of using the XMLImporter is provided by the importer demo application (ImportDemoApp.java).

Note: The XML Importer will not allow invalid data to be imported. It is your responsibility to ensure the data is valid before initiating the import process. For example, activities and resource assignments will not be imported if the actual finish date precedes the actual start date in the XML file.

If the data you would like to import is incomplete or does not conform to the full project export schema, you can still use the API to create an integration solution. One possibility would be to use DOM, SAX, or StAX to parse the XML yourself and then call the appropriate methods of the API directly. For XML files for which you have the XML Schema (XSD) available, an even better alternative might be to use an XML binding technology such as JAXB.

Note: It is highly recommended that you use the XMLImporter whenever possible due to the many complexities that can be involved in the import process in data dependency and matching.

XML Schema

- If using the P6 Professional API, the XMLExporter and XMLImporter uses the p6apibo.xsd schema.
- If using the P6 Integration API, the XMLExporter and XMLImporter uses the p6apibo.xsd schema. If the API is installed into a web/app server to support Remote Mode, the schema file can be downloaded from the web server with the URL http://<host>:<port>/PrimaveraAPI/schema/p6apibo.xsd.

Performance Tips

Many factors can affect the performance of an application that uses the API. The following tips will help the programmer avoid some of the more common performance problems:

- If possible, log in to the API as a user with the *Admin Superuser* global security profile.
- When creating or updating business objects, use methods that allow multiple objects to be processed at one time.
- When loading business objects, load only the fields that are absolutely necessary.
- Use *where* clauses to load business objects intelligently.
- When choosing the methods used to access business objects, be sure to use the methods that will most effectively minimize server and database traffic. For example, a project is the parent of its WBS hierarchy, and individual WBS objects can be parents of other business objects, such as activities. To access activities quickly, use the loadAllActivities() method of project to bypass the child WBS objects.
- Use only Local Mode with the XML exporter and importer.

Demo Applications

Demo applications are installed in the demo directory under the Integration API installation directory. Demo applications include source code, so they provide a working example of how to use the API.

The following demo applications are included with this release:

- **demo.general.GeneralDemoApp:** provides sample code for creating, loading, updating, and deleting business objects. It will, among other things, add a new project, add activities to that project, add expenses to those activities, add global and project-specific activity code types, add new activity codes for those types, and assign some activity codes to activities. It will then load the activities from the server and output them to XML.
- **demo.assignments.AssignmentsDemoApp:** loads all resource assignments across projects for the first 50 resources and generates an output HTML file. This application demonstrates the speed at which data can be accessed across projects.
- **demo.xmlexport.ExportDemoApp:** performs an XML export of the first ten projects in the database, and using the XMLExportListener interface, specifies to include all fields when exporting activities and skip activity notes and WBS milestones.
- **demo.xmlimport.ImportDemoApp:** performs an import of an XML file created using XMLExporter.exportFullProject().

For More Information

Where to Get Documentation

Complete documentation libraries for P6 EPPM releases are available from:

https://docs.oracle.com/en/industries/construction-engineering/

The documentation assumes a standard setup of the product, with full access rights to all features and functions.

Help System Access

P6 EPPM is configured to access the versions of its help systems hosted by Oracle. For on-premises, downloadable versions of the help systems are also available if you need to download, deploy, and access a local copy.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Where to Get Training

Equation 1: mylearn.oracle.com/construction

The MyLearn website provides free video-based training for all Construction and Engineering applications. On your first visit, create a free account with Oracle University and enjoy these benefits:

- More than 300 videos
- Site remembers which videos you have watched
- Filter by product
- Earn badges to share on social media
- Video captions translated into 14 languages
- Searchable transcript in English
- Build your own home page based on your preferences and favorites
- Track your progress and achievement on a personal dashboard

A variety of training is offered. (Not all training types offered for all products)

- **Get Started**: New user? These courses will get you up and running.
- **What's New**: Learn about the new features introduced in the latest release.
- **Video Training**: Single-topic, short duration videos provide instruction on basic functions and common tasks.
- **Crash Courses**: Longer-duration videos (narrated by an instructor) guide you step-by-step through processes like planning a project, or take a deep-dive into a single subject.
- **Full Virtual Courses**: Do hands-on exercises in the software and view training manuals in these comprehensive instructor-led recorded courses. Requires fee.

Where to Get Support

If you have a question about using Oracle products that you cannot resolve with information in the documentation or help:

- Visit our support website for the latest information on contacting Oracle Global Customer Support and accessing our knowledge articles: https://support.oracle.com/.
- Learn our tips and best practices for using our support services:
  - Watch the How-to Video Training Series: https://support.oracle.com/rs?type=doc&id=603505.2.
  - Read our Working Effectively With Oracle Support - Best Practices guide: https://support.oracle.com/rs?type=doc&id=166650.1.
- Access the Construction and Engineering support communities, which are moderated by Oracle and provide a place for collaboration among industry peers to share best practices: https://community.oracle.com/community/support/primavera.

Access to Oracle Support This is a legally required paragraph. Do Not Remove.

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Register on My Oracle Support

Register as a new user on My Oracle Support (MOS), so you can create Service Requests (SRs). After you register, Oracle will ask you to verify your email address. Once verified, you can log in to your MOS account.

After the first login, you need to request access to the Support Identifier (SI) number included in the welcome email you received from Oracle. This number identifies your organization's products and services and is required to use MOS.

Once you submit the SI number, you will be assigned as the Customer User Administrator (CUA) in MOS. The CUA can submit Service Requests, approve or deny user access, and assign user privileges in MOS.

Oracle recommends having at least two CUAs for every SI. Identify additional CUAs and/or authorized users who can create Service Requests on behalf of your organization. Provide them with the SI number and instruct them to register on MOS. Once they register, you will need to approve their access and assign them as a CUA in MOS.

For more information on the CUA role and/or how to complete tasks, reference the CUA for Cloud series in MOS. Note that you must have an MOS account with an SI number linked to view support content in that portal.

Creating a Service Request

To access tutorials on how to create a service request, select the "Create and Manage Service Requests" tab on our the How-to Video Training Series page: https://support.oracle.com/rs?type=doc&id=603505.2.

When you create a service request, be sure to enter the correct product information and problem details so that the request is assigned to the proper Oracle Support team.

On-premises users having issues with a related Oracle technology should contact the appropriate support line. Available technologies vary by product and include the following products:

- Oracle Access Manager
- Oracle AutoVue
- Oracle BI Publisher
- Oracle Analytics Publisher
- Oracle BPM
- Oracle Business Intelligence
- Oracle Database
- Oracle E-Business Suite
- Oracle Enterprise Manager
- Oracle Instantis EnterpriseTrack
- Oracle Server
- Oracle Value Chain Planning
- Oracle WebCenter Content Core Capabilities (formerly Universal Content Management)
- Oracle WebLogic

Using Information Centers

Information centers provide links to important support and product information. They organize documents found on My Oracle Support (MOS), providing quick access to product- and version-specific information, such as important knowledge documents, Release Value Propositions, and Oracle University training.

Visit https://support.oracle.com/rs?type=doc&id=1486951.1 to access the information center for your product.

Information centers also provide access to:

- Communities which are moderated by Oracle providing a place for collaboration among industry peers to share best practices.
- Recently published knowledge base alerts and articles.

Support Renewals Process

If it's time to renew support for your Oracle products, or if you would like to sign up for auto-renewal of your support services, visit the My Support Renewals site: https://supportrenewals.oracle.com.

Keeping Your On-Premises Software Current and Secure

To ensure you have the latest versions of your products, download and install all available patch sets from http://support.oracle.com.

To get the latest information about Critical Patch Updates, go to http://www.oracle.com/technetwork/topics/security/alerts-086861.html.