CA Workflow

API Reference Guide

r1.1.5 SP6

Contact CA

Contact CA Support

For your convenience, CA provides one site where you can access the information you need for your Home Office, Small Business, and Enterprise CA products. At http://ca.com/support, you can access the following:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Provide Feedback

If you have comments or questions about CA product documentation, you can send a message to techpubs@ca.com.

If you would like to provide feedback about CA product documentation, complete our short customer survey, which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: Introduction

This guide provides information about CA Workflow Web services APIs. This guide includes API calls that are visible through the appropriate Web Services Description Language (WSDL) when a Web service actor is defined.

There are calls for three API's:

- Worklist
- PMService
- PMService2

This section contains the following topics:

Workflow Web Service Basics (see page 9)

Workflow Web Service Basics

Workflow Web services are multithreaded applications built with Java2 and Servlet technology. Apache Jakarta Tomcat hosts these applications in the default implementation. Specifically, they are JSP and Java Servlet applications with the following platform and system requirements:

- Microsoft Windows 2000 Server
- Microsoft Windows Server 2003
- Linux or UNIX (AIX, Solaris, HP-UX)
- Web server capable of hosting Java Servlets

Process Manager Components

The Process Manager Web service is incorporated into the processmanager.war file. The Web service is automatically deployed when the WAR file is deployed.

The Process Manager Web service utilizes technology from Apache Axis. The WSDL is generated from the same base classes as the Workflow Java API.

Starting with CA Workflow r1.1, a new Process Manager API supports complex data types as part of the WSDL definition, allowing for easier processing of the data returned from various API methods. The new API also allows third party WSDL2Java tools to generate Java stub classes which are a native Java representation of input and output parameters for most methods.

Web Service Components

CA Workflow has two Web services:

- Process Manager
- Worklist

These Web services are similar in respect to implementation, security, and error handling. Divergent areas include Simple Object Access Protocol (SOAP) definitions and some data types.

Worklist Components

The Worklist Web service is incorporated into the wl-web.war file. The Web service is automatically deployed when the WAR file is deployed.

The Worklist Web service utilizes technology from Apache Axis. The WSDL is generated from the same base classes as the Workflow Java API.

How to Obtain the Services' WSDL Document

Both Web services are implemented with Apache Axis, which dynamically generates the WSDL for the service, ensuring that you have the most up-to-date information. You can view the WSDL in Internet Explorer by entering the URL for the service.

For example, to view the WSDL for the Process Manager Web service, use the following URL:

http://<servername>:CA Portal/pm/services/pmService2?wsdl

Security

As with all standard Web services, information is sent in plain text over a network using the HTTP protocol. During CA Workflow development, the W3C (World Wide Web Consortium) did not yet have a confirmed standard for Web service security. If security is an issue with your installation, we recommend using SSL as your communications protocol.

Error Handling

If an error occurs with a Web method, a SOAP Fault is returned. The SOAP Fault is the standard means of returning exception information for Web services.

The Fault message contains standardized <Message> and <Code> elements, however, the <Detail> element is the most informative. The <Detail> element contains <errorID> and <Message> elements. The <Message> contains an English string that describes errors. The <Message> elements are more suitable for developers and more appropriate messages should display for users.

Example

The following illustrates a SOAP Fault when bad parameters are supplied to LogIn() method:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <soapenv:Body>
                <soapenv:Fault>
                        <faultcode>soapenv:Server.userException</faultcode>
                        <faultstring>com.ejbtech.processmanager.services.PMServiceException:
Malformed URL</faultstring>
                        <detail>
                                <com.ejbtech.processmanager.services.PMServiceException
xsi:type="ns1:PMServiceException" xmlns:ns1="urn:services.processmanager.ejbtech.com">
                                        <errorID xsi:type="xsd:string" xsi:nil="true"/>
                                        <message xsi:type="xsd:string">Malformed
URL</message>
                                </com.ejbtech.processmanager.services.PMServiceException>
                        </detail>
                </soapenv:Fault>
        </soapenv:Body>
</soapenv:Envelope>
```

Chapter 2: The Process Manager Web Service

The Process Manager Web services are used to access the Process Manager.

The Process Manager Web Service API (PMService) in CA Workflow r1 was written as a thin layer over the internal process manager APIs, using the same HTTP RPC method as the IDE client. The API would then convert the Java objects returned by the RPC methods to XML and return the XML as a SOAP string. It was up to the client to parse these XML strings and no schema was provided for the XML types.

The new Process Manager Web Service API (PMService2) seeks to improve the original PMService in the following areas:

- Instead of returning XML as strings, the Process Manager Web Service API uses the built-in serializers provided with Apache Axis, as well as custom serializers, to return data as complex types defined in the WSDL instead of free-form XML.
- Since the Web Service API and process manager reside inside the same context in the J2EE application server, the HTTP calls will be replaced with direct calls to the ProcessManagerServer in the pm context.

A side effect of moving away from the HTTP API is that Axis must be run inside an HTTP session managed by the J2EE application server in order for clustering to work properly. This change requires the client support HTTP cookies to maintain a session with the server. Clients that do not support cookies should continue to use the original PMService API delivered with CA Workflow r1.

The old Process Manager Web Service API (PMService) is still supported in CA Workflow r1.1.

The sections that follow provide additional details and describe operations from PMService and PMService2.

This section contains the following topics:

The Process Manager Web Services Facility

To use the Process Manager Web Services Facility, use the WSDL located at http://<servername>:CA Portal/pm/services/pmService2?wsdl (or http://<servername>*:CA Portal*/pm/services/pmService?wsdl for PMService) with third-party tools to generate subclasses. These classes can then be interfaced with an application written in the appropriate language.

Process Manager Web Services Operations (PMService)

The server exposes its Process Manager interface as a Web service for interaction with third-party components. The Web service exposes management functions for dealing with definitions, activities, instances, server configuration, and so on.

An important change from CA Workflow r1 is that logInToken(String token, String pmurl) and logIn(String username, String password, String pmurl) have been deprecated; they have been replaced with logInPM(String username, String password) and logInTokenPM(String token), which log the user into the local process manager. The old methods will not work correctly when passing in a remote process manager URL.

To log in to a remote process manager, change the port address of your Web services client to that servers' PMService address.

A complete list of operations follows.

clearLog Operation

Description

Cleans up the log trace. This operation clears the Process Manager log.

Syntax

```
public void  clearLog(java.lang.String sessionID)
throws PMServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation

Return

None

Exception Thrown

PMServiceException

deleteActor Operation

Description

Deletes an actor from workflow. This method will throw an exception if there is an error removing the actor from the database, but it will not throw an exception if the actor is not found.

Syntax

```
public void deleteActor(java.lang.String sessionid, java.lang.String actorName)
throws PMServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
actorName	string	Specifies the name of the actor to remove

Return

None

Exception Thrown

PMServiceException

deleteDefinition Operation

Description

Deletes a process definition by given ID. The operation permanently removes the process definition.

Syntax

```
public void deleteDefinition(java.lang.String sessionid, java.lang.String ID)
throws PMServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
ID	string	Specifies the process definition ID

Return

None

Exception Thrown

PMServiceException

deleteInstance Operation

Description

Permanently deletes a process instance.

Syntax

```
Public void deleteInstance(java.lang.String sessionid, java.lang.String ID)
Throws PMServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
ID	string	Specifies the process instance ID

Return

None

Exception Thrown

PMServiceException

enableLogging Operation

Description

Starts and stops logging by turning logging On/Off in the Process Manager.

Syntax

```
public void enableLogging(java.lang.String sessionid, java.lang.Boolean enable)
throws PMServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
enable	Boolean	Specifies a Boolean flag that indicates if logging should be enabled

Return

None

Exception Thrown

PMServiceException

getActor Operation

Description

Returns the specified Actor. The entire Actor is returned as an XML document.

Syntax

```
public java.lang.String getActor(java.lang.String sessionid, java.lang.String actor)
throws PMServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation

Parameter	Type	Description
actor	string	Specifies the Actor name

Return

The specified Actor as an XML string

Exception Thrown

PMServiceException

getActors Operation

Description

Returns ALL the Actors. The Actors are returned as an XML document.

Syntax

```
public java.lang.String getActors(java.lang.String sessionID)
throws PMServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation

Return

All actors at the specified URL as an XML document string

Exception Thrown

PMServiceException

getApprovedEIAMUsers Operation

Description

Returns a list of comma-separated EIAM users that should be selected from an assignment action. The search results will include the first 300 users retrieved from EIAM.

Syntax

public java.lang.String getApprovedEIAMUsers (java.lang.String sessionid, java.lang.String workitemID) throws PMServiceException

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
workitemID	string	Specifies the comma-separated IDs of the workitems that should be reassigned

Return

A list of comma-separated EIAM users based on the assignment action with the specified workitems as string

Exception Thrown

PMService Exception

getApprovedEIAMUsersUsingEIAMFilters Operation

Description

Returns a list of comma-separated EIAM users that should be selected from an assignment action. The search results will include the first 300 users retrieved from EIAM.

Syntax

public java.lang.String getApprovedEIAMUsersUsingEIAMFilters (java.lang.String sessionID, java.lang.String workitemID, java.lang.String search, java..lang.String safeAttribute, java.lang.String safeOperator) throws PMServiceException

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
workitemID	string	Specifies the comma-separated IDs of the workitems that should be reassigned

Parameter	Type	Description
search	string	Specifies the value that will be used as the search text value
safeAttribute	string	Specifies the Safe Attribute value that will be used to filter search results from EIAM. Supported Safe Attributes are: A:UserName, A:FirstName, A:MiddleName, A:LastName, A:Alias, A:JobTitle, A:Company, A:Department, A:Office, A:WorkPhoneNumber, A:FaxPhoneNumber, A:MobilePhoneNumber, A:HomePhoneNumber, A:Address, A:City, A:State, A:PostalCode, A:Country, A:MailStop, A:EmailAddress, A:Description, A:IncorrectLoginCount, A:Suspended, A:GroupMembership, A:DisplayName
safeOperator	string	Specifies the Safe Operator value that will be used to filter search results from EIAM. Supported Safe Operators are: 1,2,5,6,7,8 These values correspond to EQUAL, NOTEQUAL, LESSEQUAL, GREATEREQUAL, LIKE, NOTLIKE respectively.

Return

A list of EIAM users as a string

Exception Thrown

PMServiceException

getConfiguration Operation

Description

Returns a Process Manager configuration. This returns an XML document that contains name/value pairs with configuration information for the Process Manager such as:

LogBufSize

Identifies the server log buffer size.

WebServiceTimeout

Indicates the amount of time that the Web services actor waits for a response from a Web service in seconds.

Timeout

This setting affects two back-off times:

- Specifies the wait period before retrying when no process engine is available.
- If a workitem cannot run because a process instance is suspended, specifies the time in milliseconds before it is queued again. To tune the timeout to reduce re-queuing activity, increase timeout from its default value of 3000 ms.

EnableReporting

Indicates if data is saved to the Workflow database table "stats", which is used in Workflow reports and charts. Valid values are:

- True (default indicates that the data is saved to the Workflow database.
- False indicates that the data is not saved to the Workflow database.

SMTPost

Specifies the Simple Mail Transfer Protocol (SMTP) address when the workflow sends an email.

FromName

Specifies the name that appears in the From field when the workflow sends an email.

FromAddress

Specifies the return address when the workflow sends an email.

EnginePools

Specifies the number of process engine instances that the server runs to process Workflow activities.

MaxWorkitems

Specifies the maximum number of workitems the process engine keeps in memory. Once the number of instance workitems reaches this value, they are written to disk. If your CA Workflow server has extensive memory, increasing the MaxWorkitems value speeds up the performance of large iterations.

UseSnapshots

Indicates if CA Workflow is optimized for increased throughput and better memory usage. Enter true or false in this field.

- True specifies that the CA Workflow server uses a new, memory-efficient database schema, yielding increased throughput. As a result, new process instances started from the same process definition share common information.
- False (default) specifies that the CA Workflow server uses the original database schema.

LogFileName

Specifies the file name for the Process Manager log file.

ProcessManagerURL

Specifies the URL of the Process Manager used to populate the generated WSDL.

Additional configuration information specific to other components under the Process Manager's control, such as actors, may also be present.

Syntax

```
public java.lang.String getConfiguration (java.lang.String  sessionID)
throws PMServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation

Return

The process Manager Properties as an XML document

Exception Thrown

PMService Exception

getDefinition Operation

Description

Returns the specified process definition. The entire process definition is returned as an XML document.

Syntax

```
public java.lang.String getDefinition(java.lang.String sessionID, java.lang.String ID)
throws PMServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
ID	string	Specifies the process definition ID

Return

The process definition as XML at the specified URL

Exception Thrown

PMServiceException

getDefinitionDescriptor Operation

Description

Returns a process definition descriptor for the specified process definition as an XML document.

Syntax

```
public java.lang.String getDefinitionDescriptor (java.lang.String sessionID, java.lang.String ID)
throws PMServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
ID	string	Specifies the process definition ID

Return

The process definition as XML at the specified URL

Exception Thrown

PMServiceException

getDefinitions Operation

Description

Returns an array of process definition descriptors known by the Process Manager. Each array element is an XML document representation of the process definition descriptor, rather than an entire process definition.

The format of a ProcessDefinitionDescriptor is:

n

Specifies the unique identifier of the process definition.

name

Specifies the name of the process definition.

shortname

Specifies the short name of the process definition.

description

Specifies the description of the process definition.

modified

Specifies the time, in milliseconds from midnight, January 1, 1970 UTC, when the process definition was last modified.

inuse

If set, Specifies the name of the user holding a lock on the process definition.

versions

Specifies a Java List of Version objects for each of the processes within the process definition.

Syntax

public java.lang.String[] getDefinitions (java.lang.String sessionID)
throws PMServiceException

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation

Return

An array of process definition descriptors expressed as XML documents

Exception Thrown

PMServiceException

getExternalVariables Operation

Description

Retrieves the values of a process instances external variables.

Syntax

public java.lang.String getExternalVariables (java.lang.String sessionID, java.lang.String processID)
throws PMServiceException

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
processID	string	Specifies the process instance ID

Return

The value of all external variables as an XML document

Exception Thrown

PMServiceException

getInputParametersArray Operation

Description

Returns an array of input parameters (name and type) for a specified process definition.

Syntax

public java.lang.String[]getInputParametersArray (java.lang.String sessionID, java.lang.String ID)
throws PMServiceException

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
ID	string	Specifies the process definition ID

Return

An array of strings each of which is a parameter

Exceptions

PMServiceException

getInstance Operation

Description

Returns the complete Process Instance object for the specified process instance expressed as an XML document.

Syntax

public java.lang.String getInstance (java.lang.String sessionID, java.lang.String ID)
throws PMServiceException

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
ID	string	Specifies the process instance ID

Return

The process instance at the specified URL

Exception Thrown

PMServiceException

getInstances Operation

Description

Returns an array of process instances known by the Process Manager. Each array element is an XML document that represents the ProcessInstanceDescriptor rather than an entire process instance.

The format of a ProcessInstanceDescriptor is:

id

Specifies the unique identifier of the process instance

name

Specifies the name of the process instance

description

Specifies the description of the process instance

started

Specifies the time, in milliseconds from midnight, January 1, 1970 UTC, when the process instance was started

completed

Specifies the time, in milliseconds from midnight, January 1, 1970 UTC, when the process instance was completed

status

Specifies the state of the process instance

externals

Specifies a list of the attributes of the process instance marked as external

inuse

If set, specifies the name of the user holding a lock on the process instance

Syntax

public java.lang.String[] getInstances (java.lang.String sessionID, java.lang.String definitionID, java.lang.Long from, java.lang.Long to)
throws PMServiceException

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
definitionID	string	Specifies the process definition ID
from	long	Specifies the date from which to start listing instances
to	long	Specifies the date to which instances should be listed

Return

An array of strings of ProcessInstance objects expressed as XML documents

Exception Thrown

PMServiceException

getInstanceHistory Operation

Description

Shows the instance history of a process expressed as an XML document.

Syntax

public java.lang.String getInstanceHistory (java.lang.String sessionID, java.lang.String ID, java.lang.String.history)
throws PMServiceException

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
ID	string	Specifies the process instance ID
history	string	Returns the instance history of a process expressed as an XML document

Return

The instance history of a process expressed as an XML document

Exception Thrown

PMServiceException

getInstanceImage Operation

Description

Returns a process instance status image.

Syntax

```
public java.lang.Byte[] getInstanceImage (java.lang.String sessionID, java.lang.String instanceID,
TransmissionType tp, ImageType it)
throws PMServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
instanceID	string	Specifies the process instance ID
tp	TransmissionType	A simple type enumeration that is used to specify BYTEARRAY, MIME, or DIME transmission type for image An attachment is the most efficient way to receive the instance image.
it	ImageType	A simple type enumeration that is used to specify PNG or JPEG image types

Return

If the TransmissionType is bytearray, then the image is returned as byte[] else null

Exception Thrown

PMServiceException

getInstancesByStatus

Description

Returns an array of instances for the process definition selected by status. This returns an array of process instances known by the process manager. Each array element is an XML document representing the Process Instance Descriptor, rather than an entire process instance. The format of a Process Instance Descriptor is shown under the description of getInstances.

Syntax

```
public java.lang.String  getInstancesByStatus (java.lang.String sessionID, java.lang.String definitionID,
java.lang.Long from, java.lang.Long to, java.lang.int status)
throws PMServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
definitionID	string	Specifies the process definition ID
from	long	Specifies the date from which to start listing instances
to	long	Specifies the date to which instances should be listed
status	int	Specifies the status that instances must have to be listed. See “Process Instance Status” that follows for more information

Return

An array of ProcessInstance objects expressed as XML documents

Exception Thrown

PMServiceException

getInstanceWorkItems Operation

Description

Returns all workitems associated with a particular process instance. The workitems are returned as an XML document.

Syntax

```
public java.lang.String getInstanceWorkItems (java.lang.String sessionID, java.lang.String ID,
java.lang.Boolean  showCompleted)
throws PMServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
ID	string	Specifies the process instance ID
showCompleted	Boolean	Specifies whether to return the completed workitems

Return

The workitems as an XML string at the specified URL

Exception Thrown

PMServiceException

getLog Operation

Description

Returns the entire contents of the Process Manager log file as an array of strings.

Syntax

```
public java.lang.String[]  getLog(java.lang.String sessionID)
throws PMServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation

Return

The contents of the log file as an array of strings.

Exception Thrown

PMServiceException

getResults Operation

Description

Returns process execution results as an array of attributes.

Syntax

```
public java.lang.String[] getResults (java.lang.String sessionID, java.lang.String  ID)
throws PMServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
ID	string	Specifies the process instance ID

Return

An array of string attributes; each attribute is XML

Exception Thrown

PMServiceException

getStatus Operation

Description

Returns status of the process instance.

Syntax

```
public java.lang.String getStatus (java.lang.String sessionID, java.lang.String processID)
throws PMServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation

Parameter	Type	Description
processID	string	Specifies the process instance ID

Return

The status of the process instance as a string

Exception Thrown

PMServiceException

getStatusCode Operation

Description

Returns the status code of the process instance. For a representation of the status code string, see the section “Process Instance Status” later in this chapter.

Syntax

```
public java.lang.String getStatusCode (java.lang.String sessionID, java.lang.String processID)
throws PMServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
processID	string	Specifies the process instance ID

Return

The status code as numeric value of the process instance (see the section Process Instance Status)

Exception Thrown

PMServiceException

getWSDL Operation

Description

Returns the generated WSDL for a process definition.

Syntax

public java.lang.String getWSDL (java.lang.String sessionID, java.lang.String ID)
throws PMServiceException

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
ID	string	Specifies the process definition ID

Return

The generated WSDL for the process definition

Exception Thrown

PMServiceException

isEIAMEnabled Operation

Description

Returns a Boolean that indicates if a user source is EIAM.

Syntax

public java.lang.Boolean isEIAMEnabled (java.lang.String sessionID)
throws PMServiceException

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation

Return

True if the user source is EIAM, or false if not

Exception Thrown

PMServiceException

isLoggingEnabled Operation

Description

Returns true or false to indicate whether Process Manager Server logging is enabled.

Syntax

```
public java.lang.Boolean isLoggingEnabled (java.lang.String sessionID)
throws PMServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation

Return

TRUE if logging is currently enabled or FALSE if it is not

Exception Thrown

PMServiceException

lock Operation

Description

Locks the identified process definition or process instance. The Process Manager maintains the locks in memory.

Syntax

```
public void lock(java.lang.String sessionID, java.lang.String ID)
throws PMServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
ID	string	Specifies the process definition ID

Return

None

Exception Thrown

PMServiceException

logIn Operation (Deprecated)

Description

Logs the application into the desired Process Manager. This operation has been deprecated in CA Workflow r1.1. To log in to a different process manager, supply a different port address to your Web service client and log in using the logInPM() method.

Syntax

```
public java.lang.String logIn(java.lang.String user, java.lang.String password, java.lang.String pmurl)
throws PMServiceException
```

Parameters

Parameter	Type	Description
user	string	Specifies the user ID
password	string	Specifies the password for this user
pmurl	string	Specifies the URL of the Process Manager to which the application wishes to be connected **Important!** If this value is anything other than the local PMURL, reliable operation is not guaranteed.

Return

The session ID that will be used as a handle for communicating with the Process Manager

Exception Thrown

PMServiceException

logInPM Operation

Description

Logs the user in to the local Process Manager with a username and password.

Syntax

```
public java.lang.String logInPM(java.lang.String user, java.lang.String password)
throws PMServiceException
```

Parameters

Parameter	Type	Description
user	string	Specifies the user ID
password	string	Specifies the password for this user

Return

The session ID that will be used as a handle for communicating with the Process Manager

Exception Thrown

PMServiceException

logInToken Operation (Deprecated)

Description

This operation has been deprecated in version 1.1 of CA Workflow. To login to a different process manager, supply a different port address to your Web Service client and login using the logInTokenPM() method.

Syntax

```
public java.lang.String logInToken(java.lang.String token, java.lang.String pmurl)
throws PMServiceException
```

Parameters

Parameter	Type	Description
token	string	Specifies the eTrust IAM Toolkit token
pmurl	string	Specifies the URL of the Process Manager to which the application wishes to be connected **Important!** If this value is anything other than the local PMURL, reliable operation is not guaranteed.

Return

The session ID that will be used as a handle for communicating with the Process Manager

Exception Thrown

PMServiceException

logInTokenPM Operation

Description

Logs the user in to the local Process Manager with an eIAM token

Syntax

public java.lang.String logInTokenPM (java.lang.String token)
throws PMServiceException

Parameters

Parameter	Type	Description
token	string	Specifies the eTrust IAM Toolkit token

Return

The session ID that will be used as a handle for communicating with the Process Manager

Exception Thrown

PMServiceException

logOut Operation (Process Manager)

Description

Logs out of the Process Manager session.

Syntax

public void logOut (java.lang.String sessionID)
throws PMServiceException

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation

Return

None

Exception Thrown

PMServiceException

purgeInstances Operation

Description

Permanently deletes all process instances that are created or completed from the specified process definition and run within the specified time period. The beginning and ending from and to dates are each expressed as the difference in milliseconds between that date/time and 12:00 AM, January 1, 1970 UTC (UTC is also known as GMT).

Syntax

```
public void purgeInstances (java.lang.String sessionID, java.lang.String definitionID, java.lang.Long from,
java.lang.Long  to)
throws PMServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
definitionID	string	Specifies the process definition ID
from	long	Specifies the date/time converted into milliseconds
to	long	Specifies the date/time converted into milliseconds

Return

None

Exception Thrown

PMServiceException

purgeInstancesAsynchronously Operation

Description

Permanently deletes all process instances that are created or completed from the specified process definition and run within the specified time period. The beginning and ending from and to dates are each expressed as the difference in milliseconds between that date/time and 12:00 AM, January 1, 1970 UTC (UTC is also known as GMT). This is an asynchronous operation that runs in the background.

Syntax

public void purgeInstancesAsynchronously (java.lang.String sessionID, java.lang.String definitionID, java.lang.Long from, java.lang.Long to, java.lang.Boolean completed)
throws PMServiceException

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
definitionID	string	Specifies the process definition ID
from	long	Specifies the date/time converted into milliseconds
to	long	Specifies the date/time converted into milliseconds
completed	Boolean	Specifies whether this operation should select process instances based on their start date or completion date If false, select process instances based on the start date.

Return

None

Exception Thrown

PMServiceException

purgeAsynchronouslyForCompletedWithStats Operation

Description

Permanently deletes all process instances that are created from the specified process definition and also the associated data from the STATS table and run within the specified time period. The beginning and ending from and to dates are each expressed as the difference in milliseconds between that date/time and 12:00 AM, January 1, 1970 UTC (UTC is also known as GMT). This is an asynchronous operation that runs in the background.

Syntax

public void purgeAsynchronouslyForCompletedWithStats (java.lang.String sessionID, java.lang.String definitionID, java.lang.Long from, java.lang.Long to, java.lang.Boolean completed, java.lang.Boolean deleteStats)
throws PMServiceException

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
definitionID	string	Specifies the process definition ID
from	long	Specifies the start date in number of milliseconds since January 1, 1970, 00:00:00 GMT represented by this Date object For more information, see the java.util.Date Java documentation.
to	long	Specifies the end date, which is specified in number of milliseconds since January 1, 1970, 00:00:00 GMT represented by this Date object For more information, see the java.util.Date Java documentation.
completed	Boolean	Specifies whether this operation should select process instances based on their start date or completion date If false, selects process instances based on the start date.
deleteStats	Boolean	Specifies whether this operation should delete the stats table records or not as part of purge If false, the stats table records are not deleted.

Return

None

Exception Thrown

PMServiceException

purgeInstancesSynchronously Operation

Description

Permanently deletes all process instances that are created or completed from the specified process definition and run within the specified time period. The beginning and ending from and to dates are each expressed as the difference in milliseconds between that date/time and 12:00 AM, January 1, 1970 UTC (UTC is also known as GMT). This operation returns the status of the purge operation.

Syntax

public java.lang.integer purgeInstancesSynchronously (java.lang.String sessionID, java.lang.String definitionID, java.lang.Long from, java.lang.Long to, java.lang.Boolean completed)
throws PMServiceException

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
definitionID	string	Specifies the process definition ID
from	long	Specifies the date/time converted into milliseconds
to	long	Specifies the date/time converted into milliseconds
completed	Boolean	Specifies whether this operation should select process instances based on their start date or completion date If false, select process instances based on the start date; otherwise, select based on the completed date.

Return

The purge status value (the integer and corresponding string)

Valid values are:

0 = PURGE_SUCCESS

1 = PURGE_FAILURE_INTERNAL_ERROR

2 = PURGE_LOGIN_FAILED

3 = NO_INSTANCES_TO_PURGE

Exception Thrown

PMServiceException

purgeSynchronouslyForCompletedWithStats Operation

Description

Permanently deletes all process instances that are created from the specified process definition and also the associated data from the STATS table and run within the specified time period. The beginning and ending from and to dates are each expressed as the difference in milliseconds between that date/time and 12:00 AM, January 1, 1970 UTC (UTC is also known as GMT).

Syntax

```
public void  purgeAsynchronouslyForCompletedWithStats (java.lang.String sessionID, java.lang.String definitionID, java.lang.Long from, java.lang.Long to, java.lang.Boolean completed, java.lang.Boolean deleteStats)
throws PMServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
definitionID	string	Specifies the process definition ID
from	long	Specifies the date/time converted into milliseconds
to	long	Specifies the date/time converted into milliseconds
completed	Boolean	Specifies whether this operation should select process instances based on their start or completion date If false, select process instances based on the start date.
deleteStats	Boolean	Specifies whether this operation should delete the stats table records or not as part of the purge. If false, the stats table records will not be deleted; otherwise it will delete them.

Return

The purge status value (the integer and corresponding string).

Valid values are:

0 = PURGE SUCCESS

1 = PURGE FAILURE INTERNAL ERROR

2 = PURGE LOGIN FAILED

3 = NO INSTANCES TO PURGE

Exception Thrown

PMServiceException

putActors Operation

Description

Puts actors into the Workflow database. The format of the XML file is the same as the format returned by the IDE export option or the getActors()method. Multiple actors may be present in the XML file.

Syntax

public java.lang.String[] putActors (java.lang.String sessionID, java.lang.String actorXML)
throws PMServiceException

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
actorXML	string	Specifies the XML representation of the actor/actors

Return

The Actors (an array of strings) that got inserted into the database

Exception Thrown

PMServiceException

putConfiguration Operation

escription

Sets a Process Manager configuration and replaces the server properties object used for Process Manager configuration information. The status is specified in an exception if an error occurs.

Syntax

public void putConfiguration (java.lang.String sessionID, java.lang.String[] names, java.lang.String[] values)
throws PMServiceException

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
names	string [ ]	Specifies the name of the configuration attribute that is to be set
values	string [ ]	Specifies the corresponding value for attribute

Return

None

Exception Thrown

PMServiceException

putDefinition Operation

Description

Creates a process definition and saves the specified process definition. The status is specified in an exception if an error occurs.

Syntax

```
public java.lang.String putDefinition (java.lang.String sessionID, java.lang.String definition)
throws PMServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
definition	string	Specifies the process definition expressed as an XML document

Return

The ID of the created process definition

Exception Thrown

PMServiceException

resumeInstance Operation

Description

Resumes the specified process instance, putting it back into the running state.

Syntax

public void resumeInstance (java.lang.String sessionID, java.lang.String ID)
throws PMServiceException

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
ID	string	Specifies the process instance ID

Return

None

Exception Thrown

PMServiceException

setExternalVariables Operation

Description

Sets the values of a process instances external variables. The status is specified in an exception if an error occurs.

Syntax

public void setExternalVariables (java.lang.String sessionID, java.lang.String processID, java.lang.String[] names, java.lang.String[] values)
throws PMServiceException

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
processID	string	Specifies the process instance ID
names	string [ ]	Specifies the name of the variables to set
values	string [ ]	Specifies the corresponding value for the variable

Return

None

Exception Thrown

PMServiceException

start Operation

Description

Creates a process instance. The specified process definition is instantiated (that is, a new *ProcessInstance* is created and run from the specified *ProcessDefinition*). The Process Manager chooses the version of the process definition that is the most effective. An array of parameters (name/value pairs) is also passed as an argument to this call. The status is specified in an exception if an error occurs.

Syntax

```
public java.lang.String start(java.lang.String sessionID, java.lang.String definitionID, Parameter[] parameters)
throws PMServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
definitionID	string	Specifies the process definition ID
Parameters	parameter [ ]	Specifies an array of parameters that are input to the process instance that is created The name and value fields must be set to pass a parameter to the start method of the PMService. The values for all other fields are ignored and the default values for the input parameter matching the provided name are used.

Return

The process instance ID

Exception Thrown

PMServiceException

suspendInstance Operation

Description

Suspends the specified process instance. The instance can be resumed with the resumeInstance operation. The status is specified in an exception if an error occurs.

Syntax

```
public void  suspendInstance (java.lang.String sessionID, java.lang.String ID)
throws PMServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
ID	string	Specifies the process instance ID

Return

None

Exception Thrown

PMServiceException

terminateInstance Operation

Description

Terminates the specified process instance and deletes all outstanding workitems. The status is specified in an exception if an error occurs.

Syntax

```
public java.lang.String terminateInstance (java.lang.String sessionID, java.lang.String ID)
throws PMServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
ID	string	Specifies the process instance ID

Return

None

Exception Thrown

PMServiceException

unlock Operation

Description

Unlocks a process definition or process instance identified.

Syntax

```
public void unlock (java.lang.String sessionID, java.lang.String ID)
throws PMServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
ID	string	Specifies the process definition ID

Return

None

Exception Thrown

PMServiceException

updateActor Operation

Description

Updates an Actor in CA Workflow. If multiple actors are present in the XML, only the first actor will be used. This function will throw a PMServiceException if the actor is not found in the database. The format of the XML file is the same as the format returned by the IDE export option or the getActors()method. The status is specified in an exception if an error occurs.

Syntax

public java.lang.String updateActor (java.lang.String sessionID, java.lang.String actorXML, java.lang.String actorName)
throws PMServiceException

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
actorXML	string	Specifies the new XML representation of the actor
actorName	string	Specifies the name of the actor to update

Return

None

Exception Thrown

PMServiceException

Complex Types - Process Manager Web Services

This section details the complex types used in the pmService WSDL.

Parameter Values - pmService WSDL

Parameter	Type	Description
name	string	Specifies the parameter name

Parameter	Type	Description
type	int	Specifies the parameter type Possible values are integers as follows: TYPE_STRING 0 TYPE_BYTE 1 TYPE_SHORT 2 TYPE_INTEGER 3 TYPE_LONG 4 TYPE_FLOAT 5 TYPE_DOUBLE 6 TYPE_BOOLEAN 7 TYPE_XML 8 TYPE_LIST 9 TYPE_DECIMAL 10 TYPE_DATETIME 11 TYPE_BIGINTEGER 12 TYPE_COMPLEX 100 TYPE_ARRAY 101
value	Object	Specifies the parameter value
complexTypeQName	QName	Specifies a qualified name of a complex type as defined in the XML specification
expression	Object	Specifies the expression that makes up a complex type

Process Instance Status

The following instance status values (the integer and corresponding string) are returned from the getInstance and getInstances API.

Status (int)	Status in string
0	STATUS_CREATED
1	STATUS_RUNNING
2	STATUS_COMPLETED
3	STATUS_SUSPENDED
4	STATUS_TERMINATED

Simple Types - Process Manager Web Services

TransmissionType

A simple type enumeration that is used to specify BYTEARRAY, MIME, or DIME transmission type for image.

ImageType

A simple type enumeration that is used to specify PNG or JPEG image types.

Chapter 3: The Process Manager Web Service - PMService2

PMService2 provides the same methods as the original PMService API (PMService) with the exception of the following changes:

- Instead of returning XML as strings, PMService2 will use the built-in serializers provided with Apache Axis, as well as custom serializers, to return data as complex types defined in the WSDL instead of free-form XML.
- Since the Web service API and process manager reside inside the same context in the J2EE application server, the HTTP calls will be replaced with direct calls to the ProcessManagerServer in the pm context.

This section contains the following topics:

Typical User Session

PMService1 in Axis

With PMService, a user session with the Axis client libraries might look something like this:

```
SAXReader reader = new SAXReader();
PMService service = new PmServiceServiceLocator().getpmService();
String sid = service.logInPM("caflow", "caflow");
String[] definitions = service.getDefinitions(sid);
for (int i = 0; i < definitions.length; i++) {
    Document doc = reader.read(
new ByteArrayInputStream(definitions[i].getBytes()));
    System.out.println(doc.getRootElement().elementText("name"));
}
```

PMService2 in Axis

With PMService2, the same session looks like this (note that we must explicitly tell axis to maintain the session for us):

```
PMService2 service =
new PmServiceServiceLocator().getpmService2();
((PmService2SoapBindingStub)service).setMaintainSession(true);
service.logIn("caflow", "caflow");
ProcessDefinitionDescriptor[] definitions =
service.getDefinitions();
for (int i = 0; i < definitions.length; i++) {
        System.out.println(definitions[i].getName());
}
```

PMService2 in .NET (C#)

In C#, using stubs generated by wsdl.exe, the PMService2 session would look like this (note the use of a CookieContainer to maintain the session):

```
pmServiceService service = new pmServiceService();
System.Net.CookieContainer cookies =
new System.Net.CookieContainer();
service.CookieContainer = cookies;
service.logIn("caflow","caflow");
ProcessDefinitionDescriptor[] descriptors =
service.getDefinitions();
for(int i=0;i<descriptors.Length;i++) {
        Console.WriteLine(descriptors[i].name);
}
```

Session Management

The change that will have the largest impact on how this API is used is the change to the session management code. As shown in the previous session example, PMService2 no longer returns a session ID as an explicit parameter from the login method and no longer accepts a session ID as a parameter to the other methods. Instead, Axis returns an HTTP cookie with the login method which must be passed in when any other methods are called.

The line that tells the Axis client to pass along this cookie is "((PmService2SoapBindingStub)service).setMaintainSession(true);" In .NET, adding a CookieContainer to the service instructs the generated stubs to use the cookie. No other client libraries have been tested at this time.

The reason for this change is that it allows you to use the clustering support built into application servers. With cookie-based sessions, we get a consistent and well-tested mechanism to propagate session data throughout the cluster in the case of failover, and we provide the load balancer with sufficient information to keep sessions associated with a single node in the cluster whenever possible.

Complex Types - PMService2

For all of the complex types returned from the workflow except actors, process instances, and process definitions, the schema of complex types is specified in the Web Service Definition Language (WSDL). Descriptions of the complex types for PMService2 follow.

Attribute

Field	Type	Description
complexTypeQName	String	Holds its QName when the attribute is a complex type defined in a schema, otherwise it will be null
description	String	Specifies the description of the attribute
external	Boolean	Specifies whether the attribute is marked as external
input	Boolean	Specifies whether the attribute is an input parameter
name	String	Specifies the name of the attribute
output	Boolean	Specifies whether the attribute is an output parameter
type	Integer	Specifies the type of the attribute, represented as an integer STRING = 0, BYTE = 1, SHORT = 2, INTEGER = 3, LONG = 4, FLOAT = 5, DOUBLE = 6, BOOLEAN = 7, XML = 8, LIST = 9, DECIMAL = 10, DATETIME = 11, BIGINTEGER = 12, COMPLEX = 100, ARRAY = 101
value	String	Specifies the value of the attribute

HistoryRecord

Field	Type	Description
actorName	String	Specifies the name of the actor that performed this operation
eventTypePresentation	String	Specifies the type of event, formatted as a human-readable string
info	String	Specifies information about the event
nodeId	String	Specifies the ID of the node in the definition this event is associated with
nodePresentation	String	Specifies the human-readable name of the node in the definition that this event is associated with
timestamp	Long	Specifies the java time when the event occurred
timestampPresentation	String	Specifies the human-readable time when the event occurred
Type	Integer	■ WORKITEM_CREATED = 0, ■ WORKITEM_DISPATCHED = 1, ■ WORKITEM_COMPLETED = 2, ■ WORKITEM_OVERDUE = 3, ■ WORKITEM_REMOVED = 4, ■ EXECUTED = 5, ■ STARTED = 6 ■ COMPLETED = 7 ■ EXCEPTION = 8; ■ WORKITEM_INFO = 9;
wiPresentation	String	Specifies the workitem ID as a human-readable string
workItemId	String	Specifies the workitem ID associated with the event

LogEntry

Field	Type	Description
action	String	Specifies the action this log operation occurred in

info	String	Specifies information about the log event
instanceId	String	Specifies the instance this entry pertains to, if applicable
status	Integer	Specifies whether this entry represents a success or failure
taxonomy	String	Specifies the class of the log entry
time	Long	Specifies when this log entry occurred.
workitemId	String	Specifies the workitem this log entry pertains to, if applicable

Parameter

Field	Type	Description
expression	String	Specifies the expression assigned to this parameter
name	String	Specifies the name of the parameter
type	Integer	Specifies the type of the parameter (see Attribute for a list of types)
typeQName	QName	Specifies the type of the parameter as a QName
Value	String	Specifies the value of the parameter

ProcessDefinitionDescriptor

Field	Type	Description
description	String	The description of this definition
id	String	Specifies the definition ID
inuse	String	Specifies the name of the user that has locked this definition, or null if the definition is not locked
modified	Long	Specifies the date (as a java time) when the definition was last modified
name	String	Specifies the definition name
shortName	String	Specifies the definition short name
versions	Version[]	Specifies the descriptors of all the versions of this definition

ProcessInstanceDescriptor

Field	Type	Description
completed	Long	Specifies the java time when this instance was complete, or 0 if it has not been completed
description	String	Specifies the description of this instance
externals	Attribute[]	Specifies the external attributes for the instance
id	String	Specifies the instance ID
inuse	String	Specifies the name of the user that has locked this instance, or null if the definition is not locked
name	String	Specifies the definition name associated with the instance
started	Long	Specifies the java time when this instance was started
status	Integer	Specifies the status of the instance STATUS_CREATED = 0 STATUS_RUNNING = 1 STATUS_COMPLETED = 2 STATUS_SUSPENDED = 3 STATUS_TERMINATED = 4

Version

Field	Type	Description
active	Boolean	Specifies whether this version is active
effective	Long	Specifies the first date when this version can be used (as a java time)
expiration	Long	Specifies the last date when this version can be used (as a java time)
id	String	Specifies the version ID

WorkItem

Field	Type	Description
id	String	Specifies the ID of the workitem
activated	Long	Specifies the java time when the workitem was created
actor	String	Specifies the actor that was assigned the workitem
completed	Long	Specifies the time when the workitem was completed, or 0 if the workitem was not completed
completedBy	String	Specifies the user/actor that completed the workitem
controlAttribute	String	Specifies the name of the variable that is being iterated over if this workitem is part of an iteration
controlValue	String	Specifies the value of the controlAttribute when this workitem was created
dueDate	Long	Specifies the java time when this workitem must be completed
executionId	String	Specifies the execution context the workitem belongs to This value is used in iterations.
id	String	Specifies the workitem ID
inputParameters	Parameter [ ]	Specifies the input parameters that were passed to the workitem
iteration	Integer	Specifies the place in the iteration where this workitem belongs
label	String	Specifies the label of the node this workitem represents
nodeDescription	String	Specifies the description of the node this workitem represents
nodeId	String	Specifies the ID of the node this workitem represents
nodeName	String	Specifies the name of the node this workitem represents

operation	String	Specifies the name of the operation this workitem represents
outputParameters	Parameter [ ]	Specifies the output parameters that were passed in from the actor
parentWorkItemId	String	Specifies the ID of the workitem that precedes this one
processId	String	Specifies the ID of the version of the definition this workitem belongs to
processInstanceId	String	Specifies the ID of the process instance this workitem belongs to
processName	String	Specifies the name of the definition this workitem belongs to
processValues	Parameter [ ]	Specifies any external attributes associated with the process
valid	Boolean	Specifies whether the database representation of this workitem is guaranteed to be accurate Workitems will be persisted before they are valid in the case of long iterations.
sendEmail	Boolean	Specifies whether to send an email
EmailObject	EmailObject	Specifies email details

The com.ejbtech.processmanager.services Package

placeholder for information

Class PMService2

The Process Manager Services version 2 (PMService2) provides Web service access to the process manager. Axis HTTP Cookie based session management is used. This file is meant to be used as a session scoped service. If it is used as a request or application scoped service, session management will not function correctly. This class can also be used directly in the PM context by creating a new PMService2 object and calling login.

Syntax

```
java.lang.Object
  |
  +-com.ejbtech.processmanager.services.PMService2
```

All implemented Interfaces:

```
java.io.Serializable
```

public class PMService2

extends java.lang.Object

implements java.io.Serializable

Field Summary

Field	Description
public statiC final	CACOPYRIGHT

Constructor Summary

Constructor	Description
public	PMService2()

Method Summary

Return	Operation	Description
void	clearLog()	Cleans up the log trace
java.lang.String	createDefinition(java.lang.String xml)	Creates new process definition

ActorImpl	createActor()	Creates a new actor
void	deleteActor(java.lang.String actorName)	Deletes an actor from workflow
void	deleteDefinition(java.lang.String id)	Deletes a process definition by given ID
void	deleteInstance(java.lang.String id)	Permanently removes the process instance
void	enableLogging(boolean enable)	Starts and stops logging
java.lang.String	getActor(java.lang.String actor)	Returns a single actor as an XML document
java.lang.String	getActors()	Returns all actors as an XML document
Pair[]	getConfiguration()	Returns a process manager configuration
java.lang.String	getDefinition(java.lang.String id, boolean lock)	Returns the specified process definition
ProcessDefinitionDescriptor	getDefinitionDescriptor(java.lang.String id)	Returns the Process Definition Descriptor for the specified process definition
ProcessDefinitionDescriptor[]	getDefinitions()	Returns an array of process definition descriptors
Pair[]	getExternalVariables(java.lang.String processId)	Returns the external variables for a process
Attribute[]	getInputParameters(java.lang.String id)	Returns input parameters for a process definition
java.lang.String	getInstance(java.lang.String id)	Returns string containing the complete Process Instance object for the specified process instance expressed as an XML document

java.lang.String	getInstanceDetails(java.lang.String id)	Returns string containing the "Process Instance details" for the specified process instance expressed as an XML document. The returned value contains bare-minimum details of a given process-instance compared to getInstance(String) method above, for example, in PMService2. This method does exactly what the PMService.getInstance(String) method does.
com.ejbtech.process.HistoryRecord[]	getInstanceHistory(java.lang.String id)	Returns the history of an instance as an array of HistoryRecords
byte[ ]	getInstanceImage(final String instanceid,TransmissionType tp,ImageType it)	Returns a process instance status image
ProcessInstanceDescriptor[]	getInstances(java.lang.String definitionId, long from, long to)	Returns an array of instances for the process definition
ProcessInstanceDescriptorBean[]	getInstancesByStatus()	Returns an array of instances for the process definition selected by status
com.ejbtech.processengine.WorkItem[]	getInstanceWorkItems(java.lang.String id, boolean showCompleted)	Returns all the workitems for a particular instance
LogEntry[]	getLog()	Returns the user's log
java.lang.String	getLoginName()	Returns the username associated with this session

Attribute[]	getResults(java.lang.String id)	Returns process execution results as an array of attributes
ProcessDefinitionDescriptor[]	getRunningDefinitions()	Returns an array of all the definitions that have instances in the system
java.lang.String	getStatus(java.lang.String processid)	Returns the status code of a process
java.lang.String	getStatusCode(java.lang.String processid)	Returns the status code of a process
boolean	hasDefinitionPermission(java.lang.String permissionName, java.lang.String definitionId)	Checks if the current user has a permission on a given definition
boolean	hasGlobalPermission(java.lang.String permissionName)	Checks if the current user has the given global permission
boolean	hasInstancePermission(java.lang.String permissionName, java.lang.String instanceId)	Checks if the current user has a permission on a given instance
boolean	isLoggingEnabled()	Returns true if the Process Manager Server enabled logging and false otherwise
boolean	isSessionValid()	Checks if this PMService2 session is still valid
void	lock(java.lang.String id)	Locks the process definition or process instance identified
void	logIn(java.lang.String user, java.lang.String password)	Logs in the specified user into the process manager
void	loginToken(java.lang.String token)	Logs in the specified user into the process manager using a token from eTrust IAM Toolkit

void	logOut()	Logs out of the process manager session
void	purgeAsynchronouslyForCompletedWith Stats	Permanently deletes all process instances that are created from the specified process definition and also the associated data from the STATS table and run within the specified time period
Integer	purgeSynchronouslyForCompletedWith Stats	Permanently deletes all process instances created from the specified process definition that started or completed between the specified dates and also the associated data from the STATS table and run within the specified time period
void	purgeInstances(java.lang.String definitionId, long from, long to)	Permanently deletes all process instances created from the specified process definition, and started between the dates specified
void	purgeInstancesAsynchronously	Permanently deletes all process instances created from the specified process definition that started or completed between the specified dates

Integer	purgeInstancesSynchronously	Permanently deletes all process instances created from the specified process definition that started or completed between the specified dates
java.lang.String[]	putActors(java.lang.String actorXML)	Puts actors into the workflow database
void	putConfiguration(Pair[] configuration)	Sets a process manager configuration
java.lang.String	putDefinition(java.lang.String xml)	Updates an existing process definition or creates a new process definition if the process definition id is null
void	resumeInstance(java.lang.String id)	Resumes the specified process instance, putting it back into the running state
void	setExternalVariables(java.lang.String processId, Pair[] externals)	Sets the external variables for a process
void	setLocale	Creates locale object from language and country for use in logging event messages and history
java.lang.String	start(java.lang.String definitionId, com.ejbtech.processengine.Parameter[] parameters)	Instantiates the specified process definition
void	suspendInstance(java.lang.String id)	Suspends the specified process instance
void	terminateInstance(java.lang.String id)	Terminates the specified process instance

void	unlock(java.lang.String id)	Unlocks the process definition or process instance identified
void	updateActor(java.lang.String actorXML, java.lang.String actorName)	Updates an actor in workflow

Methods Inherited for class java.lang.Object

Method
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Class PMService2 Constructors

public PMService2()

PMService2 Fields

Fields

Fields	Description
CACOPYRIGHT	public static final java.lang.String CACOPYRIGHT Example Computer Associates copyright notice. Constant value: Copyright (c) 2003-2004 Computer Associates International, Inc. All rights reserved.

Class PMService2 Methods

The methods of PMService2 mirror the methods of the original PMService.

clearLog

Description

Cleans up the log trace and clears the Process Manager log.

Syntax

```
public void clearLog()
throws PMServiceException
```

Parameters

None

Return

None

Exception Thrown

PMServiceException

createDefinition

Description

Creates a new process definition. The operation saves the specified process definition.

Syntax

```
public java.lang.String createDefinition(java.lang.String xml)
throws PMServiceException
```

Parameters

Parameter	Type	Description
xml	string	Specifies the process definition expressed as an XML document

Return

The ID of the created process definition

Exception Thrown

PMServiceException if the process definition already exists

deleteActor

Description

Deletes an actor from workflow. This method will throw an exception if there is an error removing the actor from the database, but it will not throw an exception if the actor is not found.

Syntax

```
public void deleteActor(java.lang.String actorName)
throws PMServiceException
```

Parameters

Parameter	Type	Description
actorName	string	Specifies the name of the actor to remove

Return

None

Exception Thrown

PMServiceException if there is an error removing the actor from the database

deleteDefinition

Description

Deletes a process definition by given ID and permanently removes the process definition.

Syntax

```
public void deleteDefinition(java.lang.String id)
throws PMServiceException
```

Parameters

Parameter	Type	Description
ID	string	Specifies the process definition ID

Return

None

Exception Thrown

PMServiceException

deleteInstance

Description

Permanently removes the process instance.

Syntax

```
public void deleteInstance(java.lang.String id)
throws PMServiceException
```

Parameters

Parameter	Type	Description
ID	string	Identifies the process instance ID

Return

None

Exception Thrown

PMServiceException

enableLogging

Description

Starts and stops logging performed by the Process Manager.

Syntax

```
public void enableLogging(boolean enable)
throws PMServiceException
```

Parameters

Parameter	Type	Description
enable	Boolean	A Boolean flag that indicates if logging should be enabled

Return

None

Exception Thrown

PMServiceException

getActor

Description

Returns a single actor as an XML document. This document is of the same format as the XML files saved through the export dialog in the Workflow Design Environment.

Syntax

```
public java.lang.String getActor(java.lang.String actor)
throws PMServiceException
```

Parameters

Parameter	Type	Description
actor	string	Specifies the name of the actor to retrieve

Return

An XML representation of the actor

Exception Thrown

PMServiceException

getActors

Description

Returns all actors as an XML document. This document is of the same format as the XML files saved through the export dialog in the Workflow Design Environment.

Syntax

```
public java.lang.String getActors()
throws PMServiceException
```

Parameters

None

Return

All the actors in workflow as an XML document

Exception Thrown

PMServiceException

getConfiguration

Description

Returns a Process Manager configuration. Returned values will contain name/value pairs with configuration information for the Process Manager. Some of the configuration information is:

LogBufSize

Identifies the server log buffer size.

WebServiceTimeout

Indicates the amount of time that the Web services actor waits for a response from a Web service in seconds.

Timeout

This setting affects two back-off times:

- Specifies the wait period before retrying when no process engine is available.
- If a workitem cannot run because a process instance is suspended, specifies the time before it is queued again. To tune the timeout to reduce re-queuing activity, increase timeout from its default value of 3000 ms.

EnableReporting

Indicates if data is saved to the Workflow database table "stats", which is used in Workflow reports and charts. Enter true or false in this field:

- True (default) indicates that the data is saved to the Workflow database.
- False indicates that the data is not saved to the Workflow database.

SMTPost

Specifies the Simple Mail Transfer Protocol (SMTP) address when the workflow sends an email.

FromName

Specifies the name that appears in the From field when the workflow sends an email.

FromAddress

Specifies the return address when the workflow sends and email.

Engine Pools

Specifies the number of process engine instances that the server runs to process workflow activities.

MaxWorkitems

Specifies the maximum number of workitems the process engine keeps in memory. Once the number of instance workitems reaches this value, they are written to disk. If your CA Workflow server has extensive memory, increasing the MaxWorkitems value speeds up the performance of large iterations.

UseSnapshots

Indicates if CA Workflow is optimized for increased throughput and better memory usage. Enter true or false in this field.

- True specifies that the CA Workflow server uses a new, memory-efficient database schema, yielding increased throughput. As a result, new process instances started from the same process definition share common information.
- False (default) specifies that the CA Workflow server uses the original database schema.

LogFileName

Specifies the file name for the Process Manager log file.

ProcessManagerURL

Specifies the URL of the Process Manager used to populate the generated WSDL.

Syntax

```
public Pair[] getConfiguration()
throws PMServiceException
```

Parameters

None

Return

The Process Manager properties as a map

Exception Thrown

PMServiceException

getDefinition

Description

Returns the specified process definition. The entire Process Definition is returned as a string representing an XML document.

Syntax

```
public java.lang.String getDefinition(java.lang.String id, boolean lock)
throws PMServiceException
```

Parameters

Parameter	Type	Description
ID	string	Specifies the process definition ID
lock	Boolean	Specifies whether to lock the definition

Return

The process definition at the specified URL

Throws

PMServiceException

getDefinitionDescriptor

Description

Returns the Process Definition Descriptor for the specified process definition.

Syntax

```
public ProcessDefinitionDescriptor getDefinitionDescriptor(java.lang.String id)
throws PMServiceException
```

Parameters

Parameter	Type	Description
ID	string	Specifies the process definition ID

Return

The process definition descriptor at the specified URL

Exception Thrown

PMServiceException

getDefinitions

Description

Returns an array of process definition descriptors. This returns an array of process definitions known by the Process Manager. Each array element is a ProcessDefinitionDescriptor, rather than an entire process definition.

Syntax

public ProcessDefinitionDescriptor[] getDefinitions()
throws PMServiceException

Parameters

None

Return

An array of Process Definition Descriptors

Exception Thrown

PMServiceException

getExternalVariables

Description

Returns the external variables for a process.

Syntax

public Pair[]getExternalVariables(java.lang.String *processId*)
throws PMServiceException

Parameters

Parameter	Type	Description
processid	string	Identifies the ID of the process instance

Return

An array of external variables as pairs (see Complex Types-PMServices2)

Exception Thrown

PMServiceException

getInputParameters

Description

Returns input parameters (process attributes) for a selected process definition.

Syntax

public AttributeBean[] getInputParameters(java.lang.String *id*)
throws PMServiceException

Parameters

Parameter	Type	Description
ID	string	Identifies the process definition ID

Return

A list of input parameters (see Complex Types-PMService2)

Exception Thrown

PMServiceException

getInstance

Description

Returns a string containing the complete process instance object for the specified process instance expressed as an XML document.

Syntax

public java.lang.String getInstance(java.lang.String *id*)
throws PMServiceException

Parameters

Parameter	Type	Description
ID	string	Identifies the process instance ID

Return

The process instance at the specified URL or null if no process instance was found

Exception Thrown

PMServiceException

getInstanceDetails()

Description

Returns a string containing the "Process Instance details" for the specified process instance expressed as an XML document. The returned value contains bare-minimum details of a given process-instance equal to the getInstance(String) method of PMService.

Syntax

getInstanceDetails(java.lang.String id)
throws PMServiceException

Parameters

Parameter	Type	Description
ID	string	Specifies the process instance ID

Return

The process instance at the specified URL or null if no process instance was found

Exception Thrown

PMServiceException

getInstanceHistory

Description

Returns the instance history as an array of HistoryRecords.

Syntax

public com.ejbtech.process.HistoryRecord[] getInstanceHistory(java.lang.String *id*)
throws PMServiceException

Parameters

Parameter	Type	Description
ID	string	Identifies the process instance ID

Return

An array of history records (see Complex Types-PMService2)

Exception Thrown

PMServiceException

getInstanceImage

Description

Returns a process instance status image.

Syntax

public byte[] getInstanceImage(final String instanceid, TransmissionType tp, ImageType it)
throws PMServiceException

Parameters

Parameter	Type	Description
instanceId	string	Identifies the instance id to get a status image
tp	Transmission Type	Identifies the transmission type of image which can be either byte array or attachment.
it	ImageType	Identifies the image type which can be either PNG or JPEG

Return

An instance status image either as a byte array or attachment

Exception Thrown

PMServiceException

getInstances

Description

Returns an array of instances for the process definition and returns an array of process instances known by the Process Manager. Each array element is a ProcessInstanceDescriptor rather than an entire process instance.

Syntax

public ProcessInstanceDescriptor[] getInstances(java.lang.String *definitionId*,long *from*, long *to*)
throws PMServiceException

Parameters

Parameter	Type	Description
definitionId	string	Identifies the process definition ID This parameter will be replaced by "ANY" if it is a null or empty string.
from	long	Identifies the date from which to start listing instances

to	long	Identifies the date to which instances should be listed

Return

An array of process instance objects expressed as XML documents or null if no process instances exist

Exception Thrown

PMServiceException

getInstanceWorkItems

Description

Returns all the workitems for a particular instance.

Syntax

public com.ejbtech.processengine.WorkItem[] getInstanceWorkItems(java.lang.String *id*,
boolean showCompleted)
throws PMServiceException

Parameters

Parameter	Type	Description
ID	string	Identifies the process instance ID
showCompleted	Boolean	Determines whether to include completed workitems

Return

An array of workitems (see Complex Types-PMService2)

Exception Thrown

PMServiceException

getLog

Description

Returns the user's log. This does not return the entire Process Manager log, only events that occurred since enableLogging was called.

Syntax

```
public WSLogEntry[] getLog()
throws PMServiceException
```

Parameters

None

Return

The contents of the log as an array of log entries (see Complex Types-PMService2)

Exception Thrown

PMServiceException

getLoginName

Description

Returns the user name associated with the session, which can be used to find the user name associated with a token.

Syntax

```
public java.lang.String getLoginName()
throws PMServiceException
```

Parameters

None

Return

The name of the user logged in with the session

Exception Thrown

PMServiceException

getResults

Description

Returns process execution results as an array of attributes.

Syntax

public AttributeBean[] getResults(java.lang.String *sessionId*)
throws PMServiceException

Parameters

Parameter	Type	Description
ID	string	Identifies the process instance ID

Return

An array of attributes (see Complex Types-PMServices2)

Exception Thrown

PMServiceException

getRunningDefinitions

Description

Returns an array of all the definitions that have instances in the system. This includes definitions that have only completed or terminated/suspended instances. Each array element is a ProcessDefinitionDescriptor, rather than an entire process definition.

Syntax

public ProcessDefinitionDescriptor[] getRunningDefinitions()
throws PMServiceException

Parameters

None

Return

An array of Process Definition Descriptors

Exception Thrown

PMServiceException

getStatus

Description

Returns the status of a process.

Syntax

public java.lang.String getStatus(java.lang.String *processid*)
throws PMServiceException

Parameters

Parameter	Type	Description
processid	string	Identifies the process instance ID

Return

The status of the specified process instance as string

Exception Thrown

PMServiceException

getStatusCode

Description

Returns the status code of a process. The possible status codes are:

- STATUS_CREATED = 0
- STATUS_RUNNING = 1
- STATUS_COMPLETED = 2
- STATUS_SUSPENDED = 3
- STATUS_TERMINATED = 4

Syntax

public java.lang.String getStatusCode(java.lang.String *processid*)
throws PMServiceException

Parameters

Parameter	Type	Description
processid	string	Identifies the process instance ID

Return

A process status as string for numeric value

Exception Thrown

PMServiceException

hasDefinitionPermission

Description

Checks if the current user has a permission on a selected definition. Valid definition permissions are:

- PM_EDIT_DEFINITION_SECURITY
- PM_VIEW_DEFINITION
- PM_EDIT_DEFINITION
- PM_DELETE_DEFINITION
- PM_LOCK_DEFINITION
- PM_UNLOCK_DEFINITION
- PM_START_INSTANCE
- PM_LIST_INSTANCES

Syntax

public boolean hasDefinitionPermission(java.lang.String *permissionName*,java.lang.String *definitionId*) throws PMServiceException

Parameters

Parameter	Type	Description
permissionName	string	Identifies the name of the permission to check
definitionId	string	Identifies the ID of the process definition to check against

Return

TRUE if the current user has the selected permission; otherwise, returns FALSE

Exception Thrown

PMServiceException

hasGlobalPermission

Description

Checks if the current user has the given global permission. Valid global permission names are:

- Login permissions

  PD_LOGIN

  WL_LOGIN

  WM_LOGIN

- Worklist Permissions

  WL_START_PROCESS

  WL_REASSIGN_TASK

- GlobalActorManager permissions

  AM_VIEW_ACTOR

  AM_CREATE_ACTOR

  AM_EDIT_ACTOR

  AM_DELETE_ACTOR

- GlobalProcessManager permissions

  PM_ACCESS PD_ACCESS

  PM_VIEW_CONFIGURATION

  PM_EDIT_CONFIGURATION

  PM_VIEW_LOG

  PM_CLEAR_LOG

  PM_ENABLE_LOG

  PM_LIST_DEFINITIONS

  PM_CREATE_DEFINITION

- GlobalObjectManager permissions

  OM_VIEW_OBJECT

  OM_CREATE_OBJECT

  OM_EDIT_OBJECT

  OM_DELETE_OBJECT

- Global ProcessMonitor permissions

  PMON_RUN_REPORT

  PMON_CREATE_REPORT

PMON_EDIT_REPORT

PMON_DELETE_REPORT

- Global DataType permissions

  DT_CREATE_DATATYPE

  DT_DELETE_DATATYPE

  DT_VIEW_DATATYPE

Syntax

public boolean hasGlobalPermission(java.lang.String *permissionName*)
throws PMServiceException

Parameters

Parameter	Type	Description
permissionName	string	Identifies the permission name to check

Return

TRUE if the current user has this permission; otherwise, returns FALSE

Exception Thrown

PMServiceException

hasInstancePermission

Description

Checks if the current user has a permission on a given instance. Valid instance permission names are:

- PM_VIEW_INSTANCE
- PM_EDIT_INSTANCE
- PM_DELETE_INSTANCE
- PM_SUSPEND_INSTANCE
- PM_RESUME_INSTANCE
- PM_TERMINATE_INSTANCE

Syntax

public boolean hasInstancePermission(java.lang.String *permissionName*, java.lang.String *instanceId*) throws PMServiceException

Parameters

Parameter	Type	Description
permissionName	string	Identifies the permission name to check
instanceId	string	Identifies the process instance ID to check the permission against

Return

TRUE if the current user has permission; otherwise, returns FALSE

Exception Thrown

PMServiceException

isLoggingEnabled

Description

Returns a boolean flag that indicates if logging is enabled.

Syntax

```
public boolean isLoggingEnabled()
throws PMServiceException
```

Parameters

None

Return

A boolean TRUE if the Process Manager server enabled logging; otherwise, returns FALSE

Exception Thrown

PMServiceException

isSessionValid

Description

Checks if this PMService2 session is still valid.

Syntax

```
public boolean isSessionValid()
throws PMServiceException
```

Parameters

None

Return

A boolean TRUE if the session is valid; otherwise, returns FALSE

Exception Thrown

PMServiceException

lock

Description

Locks the process definition or process instance identified. The Process Manager maintains the locks in memory.

Syntax

public void lock(java.lang.String *id*)
throws PMServiceException

Parameters

Parameter	Type	Description
ID	string	Identifies the process definition ID

Return

None

Exception Thrown

PMServiceException

logIn

Description

Logs the specified user into the Process Manager. This method is intended to be used in conjunction with session cookies to establish a user session. You must first enable session tracking on your Web services client library before calling this method.

Syntax

public void logIn(java.lang.String *user*, java.lang.String *password*)
throws PMServiceException

Parameters

Parameter	Type	Description
user	string	Identifies the user ID
password	string	Identifies the user password

Return

None

Exception Thrown

PMServiceException

loginToken

Description

Logs the specified user into the Process Manager using a token from EIAM. This method is intended to be used in conjunction with session cookies to establish a user session. You must first enable session tracking on your Web services client library before calling this method.

Syntax

```
public void loginToken(java.lang.String token)
throws PMServiceException
```

Parameters

Parameter	Type	Description
token	string	Identifies the EIAM token

Return

None

Exception Thrown

PMServiceException

logOut

Description

Logs out of the Process Manager session.

Syntax

```
public void logOut()
```

Parameters

None

Return

None

Exception Thrown

None

purgeAsynchronouslyForCompletedWithStats

Description

Permanently deletes all process instances created from the specified process definition that started or completed between the specified dates and also the associated data from the STATS table. This is an asynchronous operation that runs in the background.

Syntax

public void purgeAsynchronouslyForCompletedWithStats (java.lang.String definitionId,long from,long to, boolean completed, Boolean deleteStats)
throws PMServiceException

Parameters

Parameter	Type	Description
definitionID	string	Identifies the process definition ID
from	long	Identifies the date to which instances should be listed
to	long	Identifies the date to which instances should be listed
completed	Boolean	Specifies whether this operation should select process instances based on their start date or completion date If false, selects process instances based on the start date.
deleteStats	Boolean	Specifies whether this operation should delete the stats table records as part of purge If false, the stats table records are not deleted.

Return

None

Exception Thrown

PMServiceException

purgeInstances

Description

Permanently deletes all process instances created from the specified process definition that started between the specified dates.

Syntax

public void purgeInstances(java.lang.String *definitionId*,long *from*,long *to*)
throws PMServiceException

Parameters

Parameter	Type	Description
definitionId	string	Identifies the process definition ID
from	long	Identifies the date from which to start listing instances
to	long	Identifies the date to which instances should be listed

Return

None

Exception Thrown

PMServiceException

purgeInstancesAsynchronously

Description

Permanently deletes all process instances created from the specified process definition that started or completed between the specified dates. This is an asynchronous operation that runs in the background.

Syntax

public void purgeInstancesAsynchronously(java.lang.String definitionId,long from,long to, boolean completed)
throws PMServiceException

Parameters

Parameter	Type	Description
definitionID	string	Identifies the process definition ID
from	long	Identifies the date from which to start listing instances

to	long	Identifies the date to which instances should be listed
completed	Boolean	Specifies whether this operation should select process instances based on their start or completion date If false, selects process instances based on the start date.

Return

None

Exception Thrown

PMServiceException

purgeInstancesSynchronously

Description

Permanently deletes all process instances created from the specified process definition that started or completed between the specified dates.

Syntax

public int purgeInstancesSynchronously(java.lang.String definitionId,long from,long to, boolean completed) throws PMServiceException

Parameters

Parameter	Type	Description
definitionID	string	Identifies the process definition ID
from	long	Identifies the date from which to start listing instances
to	long	Identifies the date to which instances should be listed
completed	Boolean	Specifies whether this operation should select process instances based on their start or completion date. If false, selects process instances based on the start date.

Return

Returns the purge operation status. Valid values are:

0 = PURGE_SUCCESS

1 = PURGE_FAILURE_INTERNAL_ERROR

3 = NO_INSTANCES_TO_PURGE

Exception Thrown

PMServiceException

purgeSynchronouslyForCompletedWithStats

Description

Permanently deletes all process instances created from the specified process definition that started or completed between the specified dates and also the associated data from the STATS table and run within the specified time period.

Syntax

```
public int purgeSynchronouslyForCompletedWithStats(java.lang.String definitionId,long from,long to, boolean completed, boolean deleteStats)
throws PMServiceException
```

Parameters

Parameter	Type	Description
definitionID	string	Identifies the process definition ID
from	long	Identifies the date from which to start listing instances
to	long	Identifies the date to which instances should be listed
completed	Boolean	Specifies whether this operation should select process instances based on their start or completion date. If false, selects process instances based on the start date.
deleteStats	Boolean	Specifies whether this operation should delete the stats table records as part of purge If false, the stats table records are not deleted.

Return

Returns the purge operation status. Valid values are:

0 = PURGE_SUCCESS

1 = PURGE_FAILURE_INTERNAL_ERROR

3 = NO_INSTANCES_TO_PURGE

Exception Thrown

PMServiceException

putActors

Description

Puts actors into the workflow database. The format of the XML file is the same as the format returned by the Workflow Design Environment export option or the getActors() method. Multiple actors may be present in the XML file. If any of the actors already exist in the database, they will be skipped.

Syntax

```
public java.lang.String[] putActors(java.lang.String actorXML)
throws PMServiceException
```

Parameters

Parameter	Type	Description
actorXML	string	Identifies the XML document containing actor definitions

Return

The names of the actors that were successfully added

Exception Thrown

PMServiceException

putConfiguration

Description

Sets a Process Manager configuration. This is used to replace the server properties object used for Process Manager configuration information.

Syntax

```
public void putConfiguration(Pair[] configuration)
throws PMServiceException
```

Parameters

Parameter	Type	Description
configuration	pair[]	Specifies a map containing the name, and value pairs for the properties

Return

None

Exception Thrown

PMServiceException

putDefinition

Description

Updates an existing process definition or creates a new process definition if the process definition id is null.

Syntax

```
public java.lang.String putDefinition(java.lang.String xml)
throws PMServiceException
```

Parameters

Parameter	Type	Description
xml	string	Specifies the process definition expressed as an XML document

Return

The ID of the updated or created process definition

Exception Thrown

PMServiceException

resumeInstance

Description

Resumes the specified process instance, putting the instance back into a running state.

Syntax

public void resumeInstance(java.lang.String *id*)
throws PMServiceException

Parameters

Parameter	Type	Description
ID	string	Identifies the process instance ID

Return

None

Exception Thrown

PMServiceException

setExternalVariables

Description

Sets the external variables for a process instance.

Syntax

public void setExternalVariables(java.lang.String *processId*,
Pair[] externals)
throws PMServiceException

Parameters

Parameter	Type	Description
processId	string	Identifies the ID of the process instance to modify
attrMap	pair[]	Identifies an array of name/value pairs (see Complex Types-PMServices) If a name does not match an external variable for the process, the name is ignored. All names and values should be of the type String.

Return

None

Exception Thrown

PMServiceException

start

Description

Starts a new process instance from the specified process definition. The Process Manager chooses the version of the process definition that is the most effective.

Syntax

public java.lang.String start(java.lang.String *definitionId*,
com.ejbtech.processengine.Parameter[] *parameters*)
throws PMServiceException

Parameters

Parameter	Type	Description
definitionId	string	Identifies the process definition ID
parameters	parameter[]	Identifies an array of parameters that are input to the process instance that is created

Return

The process instance ID

Exception Thrown

PMServiceException

suspendInstance

Description

Suspends the specified process instance. The process instance can be resumed with the resumeInstance operation.

Syntax

public void suspendInstance(java.lang.String *id*)
throws PMServiceException

Parameters

Parameter	Type	Description
ID	string	Identifies the process instance ID

Return

None

Exception Thrown

PMServiceException

terminateInstance

Description

Terminates the specified process instance. All outstanding workitems are deleted.

Syntax

public void terminateInstance(java.lang.String *id*)
throws PMServiceException

Parameters

Parameter	Type	Description
ID	string	Identifies the process instance ID

Return

None

Exception Thrown

PMServiceException

unlock

Description

Unlocks the process definition or process instance identified.

Syntax

public void unlock(java.lang.String *id*)
throws PMServiceException

Parameters

Parameter	Type	Description
ID	string	Identifies the process definition ID

Return

None

Exception Thrown

PMServiceException

updateActor

Description

Updates an actor in workflow. The format of the XML is the same as the format returned by the export actor function of the IDE or the getActor method of this API. If multiple actors are present in the XML, only the first actor will be used. This function will throw a PMServiceException if the actor is not found in the database.

Syntax

```
public void updateActor (java.lang.String actorXML, java.lang.String actorName)
throws PMServiceException
```

Parameters

Parameter	Type	Description
actorXML	string	Specifies the new XML representation of the actor
actorName	string	Specifies the name of the actor to update

Return

None

Exception Thrown

PMServiceException if the actor is not found, or if there is error parsing the XML or storing it in the database

Chapter 4: The Worklist Web Services Facility

Using Worklist Web services, a customer can create their own custom client program to interface with CA Workflow. When you want to replace/complement human actors with software that shares work with them, the Worklist Web services can provide an alternative to using the Worklist Client UI.

For example, a fax actor could be an application most of the time but, when the fax server is not running, the workitems could be done by an individual.

The Worklist Web service operations are identical to the RPC calls used by the Worklist Client application.

This section contains the following topics:

Using the Worklist Web Services Facility

To use the Worklist Web services facility, use the WSDL located at http://<*servername*>:CA Portal/wl/services/wlService?wsdl with third-party tools to generate stub classes. These classes can then be interfaced with an application written in the appropriate language.

Worklist Web Services Operations

The server exposes its Worklist interface as a Web service for interaction with third-party components. The Web service exposes functions for detailed management of activities and workitems.

An important change from CA Workflow r1 is that logInToken(String token, String wlurl) and logIn(String username, String password, String wlurl) have been deprecated. The operations have been replaced with logInWL(String username, String password) and logInTokenWL(String token), which log the user into the local worklist. The old methods will not work correctly when passing in a remote worklist URL.

To log in to a remote worklist

Change the port address of your Web services client to that servers WLService address.

Summary of operations

A complete list of operations follows.

completeActivity Operation

Description

Completes the specified workitem identified by the worklitemID. If the actor operation that was to be performed for this workitem assigns values to output parameters (such as the query user operation) then these values must be passed in to the completeActivity operation as a comma-separated list. These values should be ordered in the way they appear in the output parameters dialog in the Workflow Design Environment. If the number of values does not match the number of output parameters for the workitem, then a WLServiceException will be thrown.

Syntax

```
public java.lang.Long  completeActivity (java.lang.String sessionID, java.lang.String dialogParam,
java.lang.String workitemID)
throws  WLServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
dialogParam	string	Specifies a comma-separated list of parameter values to complete the query user operation
workitemID	string	Specifies the ID of the workitem to be completed

Return

The completion date as long

Exception Thrown

WLServiceException

getApprovedJNDIUsers Operation

Description

Returns the list of comma-separated LDAP users that should be selected from in an assignment action.

Syntax

```
public java.lang.String getApprovedJNDIUsers (java.lang.String sessionID,java.lang.String workitemID)
throws  WLServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
workitemID	string	Specifies the comma-separated IDs of the workitems that should be reassigned

Return

The list of LDAP users

Exception Thrown

WLServiceException

getInputParameters Operation

Description

Returns the input parameters for a Workitem. Each input parameter returned is separated by a semicolon and is made up of the following three values separated by commas:

- The parameter name
- The parameter type
- The parameter value

Syntax

public java.lang.String getInputParameters (java.lang.String sessionID, java.lang.String workitemID) throws WLServiceException

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
workitemID	string	Specifies the ID of the workitem

Return

The input parameters of the Workitem specified in the following format:

<name>,<type>,<value>;...;<name>,<type>,<value>

Exception Thrown

WLServiceException

getInputParametersNumber Operation

Description

Returns the number of input parameters for a workitem.

Syntax

public java.lang.String getInputParametersNumber (java.lang.String sessionID, java.lang.String workitemID) throws WLServiceException

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
workitemID	string	Specifies the ID of the workitem

Return

The number of input parameters as string

Exception Thrown

WLServiceException

getOutputParametersNumber Operation

Description

Returns the number of output parameters for a workitem.

Syntax

public java.lang.String getOutputParametersNumber (java.lang.String sessionID, java.lang.String workitemID)
throws WLServiceException

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
workitemID	string	Specifies ID of the workitem

Return

The number of output parameters as a string

Exception Thrown

WLServiceException

getProcessAttributes Operation

Description

Returns the list of process attributes that have been marked as external. Each process attribute entry is separated by a semicolon and is made up of three values separated by commas:

- attribute name
- attribute type (the value for type is lower case such as “string” and “integer”)
- attribute value

Syntax

public java.lang.String getProcessAttributes (java.lang.String sessionID, java.lang.String workitemID)
throws WLServiceException

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation

Parameter	Type	Description
workitemID	string	Specifies the ID of the workitem

Return

The list of process attributes

Exception Thrown

WLServiceException

completeForm Operation

Description

Completes the specified form workitem identified by the workitemID.

Syntax

```
public java.lang.Long completeForm (java.lang.String sessionID, java.lang.String dialogParam,
java.lang.String workitemID)
throws  WLServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
dialogParam	string	Specifies a semicolon-delimited list of name/value pairs to complete the form **Note:** The values can be a comma delimited list. Sample string: company=CA;group=sales;members=joe, bob, eric
workitemID	string	Specifies the ID of the workitem to be completed

Return

The completion date as long

Exception Thrown

WLServiceException

getWorkItems Operation

Description

Returns the workitems for all users, including workitems for groups.

Syntax

```
public java.lang.String getWorkItems (java.lang.String sessionID)
throws  WLServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation

Return

An XML document that represents all workitems

Exception Thrown

WLServiceException

getWorkItemsForActor Operation

Description

Returns the workitems assigned to an actor excluding the groups to which this actor belongs. An XML document containing all of the workitems assigned to a user is returned.

Syntax

```
public java.lang.String getWorkItemsForActor (java.lang.String sessionID)
throws  WLServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation

Return

The XML string of workitems assigned to the actor

Exception Thrown

WLServiceException

isLDAPDirEnabled Operation

Description

Returns a Boolean that indicates if a user source is an LDAP directory.

Syntax

```
public java.lang.Boolean isLDAPDirEnabled (java.lang.String sessionID)
throws  WLServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation

Return

Returns TRUE if the user source is an LDAP directory, or FALSE if not

Exception Thrown

WLServiceException

logIn Operation (Deprecated)

Description

Logs the user into the desired worklist session. This method has been deprecated and replaced with logInWL(String username, String password). If wlUrl is any value other than the local worklist URL, then correct operation is not guaranteed, especially in a clustered environment.

Syntax

```
public java.lang.String logIn (java.lang.String user, java.lang.String  password, java.lang.String  url)
throws  WLServiceException
```

Parameters

Parameter	Type	Description
user	string	Specifies the user ID
password	string	Specifies the password for this user
wlUrl	string	Specifies the URL of the worklist session to which the application wishes to be connected

Return

The session ID that will be used as a handle for communicating with the worklist

Exception Thrown

WLServiceException

logInToken Operation (Deprecated)

Description

Logs the user into the desired worklist session. This method has been deprecated and replaced with logInTokenWL(String token). If wlUrl is any value other than the local worklist URL, then correct operation is not guaranteed, especially in a clustered environment.

Syntax

```
public java.lang.String logIn (java.lang.String  token, java.lang.String url)
throws  WLServiceException
```

Parameters

Parameter	Type	Description
token	string	Specifies the eTrust IAM Toolkit token
wlUrl	string	Specifies the URL of the worklist session to which the application wishes to be connected

Return

The session ID that will be used as a handle for communicating with the worklist

Exception Thrown

WLServiceException

logInTokenWL Operation

Description

Logs the user into the local worklist.

Syntax

public java.lang.String logInTokenWL(java.lang.String token)
throws WLServiceException

Parameters

Parameter	Type	Description
token	string	Specifies the eTrust IAM Toolkit token

Return

The session ID that will be used as a handle for communicating with the worklist

Exception Thrown

WLServiceException

logInWL Operation

Description

Logs the user into the local worklist.

Specifies a semicolon-delimited list of name/value pairs to complete the form

Note: The values can be a comma delimited list.

Sample string: company=CA;group=sales;members=joe, bob, eric

Syntax

public java.lang.String logInWL (java.lang.String usename, java.lang.String password)
throws WLServiceException

Parameters

Parameter	Type	Description
username	string	Specifies the user ID
password	string	Specifies the password for this user

Return

The session ID that will be used as a handle for communicating with the worklist

Exception Thrown

WLServiceException

logOut Operation (Worklist)

Description

Logs out of the desired worklist session.

Syntax

```
public void logOut (java.lang.String sessionID)
throws WLServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation

Return

None

Exception Thrown

WLServiceException

makeAssignment Method

Description

Returns the string SUCCESS if the assignment action was successful, or returns a list of coma separated rejection messages. Use the makeAssignment method to take, return, delegate, or reassign workitems. Take and return actions are used when working with group workitems. To perform a group workitem, a user must first take it from the group list. A workitem that was taken from a group can be returned to the group.

The delegate action is used to delegate the execution of a workitem to another user. A duplicate of the delegated workitem will appear in that user's worklist; when the user performs the workitem, both are marked as completed.

The reassign action is used to reassign a workitem to another actor (user or group).

Multiple workitems can be acted on by this operation. The parameters selectedWIs, selectedWINames, selectedUsersTo, and selectedTaskOwners list the workitems that are to be acted on, the names of these workitems, to whom the workitems are to be assigned, and the owners of these workitems.

The parameters assignedby, completedbefore, and dateformat are used only if the action is delegated. The assignedby parameter is the user who is delegating the workitems. The completedbefore and dateformat parameters allow the user who is delegating to provide a due date for the delegated workitem.

Syntax

public java.lang.String makeAssignment (java.lang.String sessionID, java.lang.String assignmentAction , java.lang.String selectedWIs, java.lang.String selectedWINames, java.lang.String selectedUsersTo, java.lang.String selectedTaskOwners , java.lang.String assignedby, java.lang.String completedbefore,java.lang.String dateformat)
throws WLServiceException

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation
assignmentAction	string	Specifies the assignment action Valid actions are: take return delegate reassign
selectedWIs	string	Specifies a comma-separated list of workitem IDs
selectedWINames	string	Specifies a comma-separated list of corresponding workitem names
selectedUsersTo	string	Specifies a comma-separated list of userids to whom the corresponding workitems are to be assigned For the reassign action, only a single userid can be specified.

Parameter	Type	Description
selectedTaskOwners	string	Specifies a comma-separated list of workitem owners corresponding to the list of workitems
assignedby	string	Specifies the name of the user who is assigning the workitems Used for delegation only.
completedbefore	string	Specifies the date by which a workitem must be completed Used for delegation only.
dateformat	string	Specifies the date format used in the completedbefore parameter

Return

"SUCCESS" or a comma-separated list of rejection messages

Exception Thrown

WLServiceException

turnOffAutoDelegation Operation

Description

Turns off the AutoDelegation setting in the worklist for the currently logged in user.

Syntax

```
public void  turnOffAutoDelegation (java.lang.String  sessionID)
throws WLServiceException
```

Parameters

Parameter	Type	Description
sessionID	string	Specifies the session ID returned by the login operation

Return

None

Exception Thrown

WLServiceException

turnOnAutoDelegation Operation

Description

Turns on the auto delegation setting in the worklist by specifying which user or group the workitems will be auto delegated to.

Syntax

```
public void  turnOnAutoDelegation (java.lang.String  sessionID, java.lang.String  autoDelegate)
throws WLServiceException
```

Parameters

Parameter	Type	Description
SessionID	string	Specifies the session ID returned by the login operation
autoDelegatedTo	string	Specifies the user name or group name to whom the workitems will be auto delegated to

Return

None

Exception Thrown

WLServiceException

Complex Types - Worklist Web Services

This section details the complex types that are used in the wlService WSDL.

WorkItem - wlService WSDL

Parameters

Field	Type	Description
expression	string	Specifies the expression assigned to this parameter
name	string	specifies the name of the parameter
type	Integer	Specifies the type of the parameter (see Attribute for a list of types)
typeQName	QName	Specifies the type of the parameter as a QName
xalue	string	Specifies the value of the parameter

Parameter	string	Description
id	string	Specifies the ID of the workitem
parentWorkItemId	string	Specifies the ID of the parent workitem
processInstanceId	string	Specifies the ID of the process instance to which this workitem belongs
processId	string	Specifies the ID of the process definition version which has been instantiated
processName	string	Specifies the name of the process
label	string	Specifies the value of the activity label attribute
processValues	Parameter [ ]	Specifies an array of process attributes and their values These are the process attributes that have been marked as external.
nodeId	string	Specifies the ID of the node for which this workitem has been created
nodeName	string	Specifies the name of the node
nodeDescription	string	Specifies the contents of the node description field This can be used to provide additional information to the actor.

Parameter	string	Description
executionId	string	
iteration	int	In the case of an iteration node, contains the iteration number
controlAttribute	string	In the case of an iteration node, specifies the name of the variable that is being iterated over if this workitem is part of an iteration
controlValue	string	In the case of an iteration node, specifies the value of the controlAttribute when this workitem was created
operation	string	Specifies the operation that is to be performed
activated	long	Specifies the date when this workitem was created and activated
completed	long	Specifies the date when this workitem was completed
dueDate	long	Specifies the date when this workitem is due
inputParameters	Parameter [ ]	Specifies an array of input parameters and their values
outputParameters	Parameter [ ]	Specifies an array of output parameters that must be returned by the actor
actor	string	Specifies the actor name
valid	boolean	Specifies whether this workitem is valid
completedBy	string	Specifies who completed the workitem
sendEmail	boolean	Specifies whether to send an email
EmailObject	EmailObject	Specifies email details

Parameter Values - wlService WSDL

Parameter	Type	Description
name	string	Specifies the parameter name

Parameter	Type	Description
type	int	Specifies the parameter type Possible values are: TYPE_STRING TYPE_BYTE TYPE_SHORT TYPE_INTEGER TYPE_LONG TYPE_FLOAT TYPE_DOUBLE TYPE_BOOLEAN TYPE_XML TYPE_LIST TYPE_DECIMAL TYPE_DATETIME TYPE_BIGINTEGER TYPE_COMPLEX TYPE_ARRAY
value	Object	Specifies the parameter value
complexTypeQName	Qname	Specifies a qualified name of a complex type as defined in the XML specification
expression	Object	Specifies the expression that makes up a complex type

Chapter 5: Exceptions Thrown

Exceptions are defined in CA Workflow components.

This section contains the following topics:

Summary of Exceptions Thrown

The PMServiceException and WLServiceException listed in the following table work as exception transmitters. Both classes are wrapped around the CallerException in the form of a normal JavaBean (http://java.sun.com/products/javabeans * mergeformat) according to the JAX-RPC specification.

Exception	Package
ActorException	com.ejbtech.actormanager
ActorExportNotSupportedException	com.ejbtech.actormanagerri
ActorFaultException	com.ejbtech.actormanagerri
ActorImportException	com.ejbtech.actormanagerri

Exception	Package
ActorManagerException	com.ejbtech.actormanager
CallerException	com.ejbtech.security
DataTypeManagerException	com.ejbtech.datatypemanager
FormException	com.ejbtech.worklist.forms
LoginException	javax.security.auth.login
ObjectClassException	com.ejbtech.objectmanager
ObjectExportNotSupportedException	com.ejbtech.objectmanagerri
ObjectImportException	com.ejbtech.objectmanagerri
ObjectManagerException	com.ejbtech.objectmanager
PMInterfaceException	com.ejbtech.processdesigner
PMServiceException	com.ejbtech.processmanager.services
ProcessExecutionException	com.ejbtech.processengine
ProcessManagerException	com.ejbtech.processmanager
SecurityManagerException	com.ejbtech.security
TypeConversionException	com.ejbtech.util
UnsupportedCallbackException	javax.security.auth.callback
WLServiceException	com.ejbtech.worklist.services

ActorException

There are two types of an ActorException:

- Thrown on design time
- Thrown on runtime

ActorExportNotSupportedException

An ActorExportNotSupportedException is thrown to indicate that an actor does not support the export operation.

ActorFaultException

An ActorFaultException is a runtime exception to signal an actor operation failure.

ActorImportException

An ActorImportException is thrown to indicate that an actor does not support the import operation.

ActorManagerException

An ActorManagerException is thrown to indicate one of the following:

- A user does not have permission to execute one of the Actor Manager functions, or when a security error occurs
- The Actor Manager function throws an exception

CallerException

A CallerException is thrown when an HTTP call to any of the system-defined APIs returns an error.

DataTypeManagerException

A DataTypeManagerException is thrown to indicate one of the following:

- A user does not have permission to execute one of the Data Type Manager functions, or when a security error occurs
- The Data Type Manager function throws an exception

Design Time ActorException

Indicates the following:

- The name of the actor created is blank or cannot be evaluated
- One of the mandatory fields in an actor property dialog does not contain data or the data it contains cannot be evaluated
- An attempt to create an actor with a name that already exists
- An attempt to create a service provider with a name that already exists

FormException

A FormException is thrown to indicate a failure in building a WEB Form for the worklist. An exception can be a failure to parse or format input data, or a missing parameter or workitem.

LoginException

A LoginException is thrown when a user provides the incorrect credentials to log in to an application.

ObjectClassException

An ObjectClassException is a runtime exception for the ObjectClass and indicates an operational failure, for example, assigning or completing a workitem.

ObjectExportNotSupportedException

Indicates that an ObjectClass does not support export operation.

ObjectImportException

Indicates that an ObjectClass does not support import operation.

ObjectManagerException

Indicates one of the following:

- A user does not have permission to execute one of the Object Manager functions or any of security errors occur
- An Object Manager function throws an exception

PMInterfaceException

Works as a wrapper for the client presentation of the ProcessManagerException and is thrown to indicate that there was an attempt to open, save, unlock, or delete a definition, open an instance, or update a security predicate.

ProcessExecutionException

Indicates one of the following:

- An attempt to load an already terminated process; an attempt to complete an activity, object, or an event (send, timer, or due date event) of a process that is terminated
- An attempt to start a process that was not created
- An attempt to start a process with input parameters that are not set
- An attempt to suspend a process that is not running
- An attempt to resume a process that is not suspended
- An attempt to resume a process with resume instructions that are not found
- An attempt to fire a timer event for a timer node that is not found
- An attempt to terminate a completed process; an attempt to complete an activity, object, or event (send, timer, or due date event) of a process that is completed
- An attempt to complete an activity or object, mark an activity incomplete, fire a send, timer, or due date event, add a work item history record, or stop a subactivity and correspondent workitem cannot be found
- An attempt to get or set the value of a process attribute that cannot be found
- An attempt to create a wait or send event work item with an undefined operation
- An attempt to complete an activity, mark an activity incomplete, fire a due date event, process exception, and correspondent activity workitem does not have an associated actor
- An attempt to get an actor for a role that has no actor assigned
- An attempt to handle an object operation on an object that is not found
- An attempt to handle an object operation on an object defined on a class that is not found
- When call to an actor to complete activity, mark activity completed, or incomplete returns an error
- A process has an unsupported attribute type
- An attempt to complete a work item that is invalid
- An attempt to complete a work item that is already completed
- An attempt to set a process attribute that cannot be found
- An error occurred during evaluating an expression
- An actor that must be configured is not configured

ProcessManagerException

Defines one of the following exceptions:

- Process engine originated

  The ProcessManagerException wraps a ProcessExecutionException

- Process manager originated

  Indicates an operation error on a process instance or process definition, such as an error that results from saving or reading a process instance, or deleting or locking a process definition.

- Security manager originated

  Indicates that no permission has been granted for the current user to execute a ProcessManager operation.

Runtime ActorException

A runtime ActorException indicates a failure of operation that an actor defined.

SecurityManagerException

A SecurityManagerException is thrown to indicate a communication error during the smapi call or a missing definition security predicate.

TypeConversionException

Indicates that there is an exception of type conversion thrown. Type conversion is used to get values of process attributes or operation parameters for supported types.

UnsupportedCallbackException

An UnsupportedCallbackException is thrown when the CallbackHandler defined to retrieve specific authentication data does not recognize a particular Callback. (A *Callback* is an object that contains the information requested to be retrieved.)

Chapter 6: Sample Web Service Workflow

This CA Workflow sample prompts the user for the ISBN number (International Standard Book Number) of a book, calls a Web service to check the price of the book, and then either approves or denies the request based on the returned price (the Web service returns a random price regardless of the input ISBN). Finally, the workflow alerts the user that the request was either approved or denied.

Note: Before following this example, contact your system administrator to receive a valid user ID to access the Workflow Design Environment.

This section contains the following topics:

Workflow Steps

1. Get ISBN

   This activity prompts the worklist user for the ISBN of the book that is requested and stores it in the attribute ISBN.

   For the purposes of this example, enter any fictional ISBN number.

2. Get Book Price

   This activity queries a Web service for the price of the book.

   For this example, the Web service generates a random price.

3. Approved Message

   This activity informs the worklist user that the request was approved.

   This activity is generated only if the price is under $50.00.

4. Denied Message

   This activity informs the worklist user that the request was denied.

   This activity is generated only if the price is $50.00 or higher.

Roles Used

User

The user who is requesting the book.

Web service

The Web service that returns the price of the book.

Set Up the Workflow

The following steps are used in the sample Web Service workflow. Refer to the corresponding topics that follow for more information about each of these steps:

1. Deploy the sample Web Service (see page 126)
2. Set up an actor for the user role (see page 126)
3. Create a JNDI User Actor when you are using LDAP authentication (see page 127)
4. Set up a web service for the WebService role (see page 127)
5. Import the WebService actor (see page 127)
6. Add the WebService actor manually (see page 128)
7. Import the workflow (see page 128)
8. Execute the workflow (see page 129)

Deploy the Sample Web Service

To deploy the sample Web Service

1. Copy the sample bookprice-webservice.war file into the webapps directory of the Workflow Tomcat application server.

   By default, the webapps directory can be found in the following location:

   c:\Program Files\CA\CA Workflow\jakarta-tomcat-4.1.31\webapps

2. You must restart the Tomcat server after copying this war file into its webapps directory.

Set Up an Actor for the User Role

The instructions that follow show you how to set up an actor for the User role according to the type of authentication you are using.

Steps That Apply When You Are Using LDAP Authentication

Note: If you already have an existing JNDI user actor, you can continue to the next section; otherwise, follow these steps to add one.

To manually create a JNDI User actor

1. Click the Actors tab on the left side of the Workflow Design Environment.
2. Select JNDI User Actor from the tree on the left.
3. Right-click JNDI User Actor, then choose Add Actor.
4. Enter **caflow** as the JNDI Actor Name.
5. Enter other LDAP parameters on this screen as appropriate for your LDAP environment.
6. Click OK.

Set Up a Web Service for the WebService Role

You can import the WebService actor from the provided XML or create the actor manually.

Note: If your Workflow Tomcat server is running on a different machine than your Workflow Design Environment, or if it is running on a port other than 8081, you must create the actor manually and edit the WSDL URL accordingly. (Importing the sample WebService actor cannot be done under these circumstances because the WebService actor is created with a WSDL URL that points to http://localhost:8081.)

Import the WebService Actor

To import the WebService actor

1. Choose File, Import, Actor from any tab in the Workflow Design Environment.
2. Select the BookPrice.xml file from the directory you unzipped this example to, then click Open.

   Note: The sample BookPrice.xml is also shown later in this chapter.

   Check the output in the Import Actors window to make sure that the actor was imported successfully, then click Close.

   Note: If your port number is different from the default (8081), a Premature end of file error will display. You must add the Actor manually.
3. Expand the WebService node of the tree in the Actors tab.

   You should now see a WebService actor called BookPrice with a single operation called getPrice.

Add the WebService Actor Manually

To add a WebService actor manually

1. Click the Actors tab in the Workflow Design Environment.
2. Right-click WebService, then choose Add Actor.
3. Enter http://*%SERVERNAME%:%PORT%*/bookprice-webservice/services/BookPrice?wsdl in the WSDL URL field.

   where:

   %SERVERNAME% is the machine name of the workflow server.

   %PORT% is the port number to connect to the workflow server.

   For example, you can enter **localhost:8081** for %SERVERNAME%:%PORT%**.**

   Note: If you are connecting to a remote machine or using a port other than 8081, you must change the URL to reflect that.
4. Enter **BookPrice** in the Name field.

   The Activity Operations tab in the right pane should now contain one activity called getPrice.

Import the Workflow

To import the workflow

1. Choose File, Import, Process Definition from the Workflow Client Process Manager tab.
2. Select WebServiceExample.xml.
3. Double-click the Web Service Example workflow that appears in the right pane.

   You are now in the Process Designer.
4. Double-click each role in the bottom pane and associate them with the actors you have created.
   - The Webservice role should be associated with the BookPrice WebService actor.
   - The User role should be associated with the caflow Users actor.
5. Click Save to save the Web Service Example process.

Execute the Workflow

If you have followed the previous instructions, everything is now set up for the Web Service Example workflow.

To execute the workflow

1. Select the Web Service Example workflow from the Process Manager tab.
2. Click the Run Process button to start the workflow.

   You should see the “Successfully started process” message.
3. Click OK.
4. Log in to your worklist (http://*%SERVERNAME%:%PORT%*/wl).

   If you are using LDAP authentication, log in as a user who is part of the LDAP list of users who were included in the filter specified during the JNDI User Actor configuration.

   For example, enter **localhost:8081** for %SERVERNAME%:%PORT%**.**

   You should see a Get ISBN activity in your worklist.
5. Click Perform.
6. Enter any value in the ISBN field (any fictional number to represent an ISBN number can be used), then click OK.
7. Click Refresh.

   You should see an activity named either Approved Message or Denied Message that states the request was either approved or denied. The price of the book also displays.
8. Click Perform for the Approved Message or Denied Message activity.

   You will see a message that the Book Order was either approved or denied, along with its cost.
9. Click OK.

   You should see that the Get ISBN and Approved Message (or Denied Message) tasks have been completed.

Web Service Workflow Example Files

The following Web Service Workflow Example files are referenced in this section and are located in a file named Web Service Workflow.zip on the Workflow installation CD in the Examples directory.

- WebServiceExample.xml
- BookPrice.xml
- caflow.xml
- bookprice-webservice.war

Chapter 7: Sample PMService Client

This section contains the following topics:

Sample PMService Client that Reads an Instance Image MIME Attachment

This PMService Client example uses a MIME attachment when a getInstanceImage call is provided.

```
package com.ca.workflow.pmservice.client;

import javax.activation.DataHandler;
import javax.xml.rpc.ServiceException;

import com.ejbtech.processmanager.services.PMServiceException;
import com.ejbtech.processmanager.services.PMService;
import com.ejbtech.processmanager.services.PmServiceServiceLocator;
import com.ejbtech.processmanager.services.PmServiceSoapBindingStub;

public class InstanceImageReader{

        private PMService service;
        private String sessid;

        private void init(String pmServiceUrl) throws Exception{
                PmServiceServiceLocator locator = new PmServiceServiceLocator();
                service = locator.getpmService(new java.net.URL(pmServiceUrl));
        }

        private void login(String userid, String passwd,
                                        String pmURL) throws Exception{
                sessid = service.logIn(userid, passwd,pmURL);

        }

        public      void getInstanceImage( String pmUrl,
                                                String pmServiceUrl,
                                                String userid,
                                                String passwd,
                                                String instanceId,
                                                String imgFilePath) throws Exception{
                init(pmServiceUrl);
                login(userid,passwd,pmUrl);
                //Transmission type is MIME attachment
                com.ca.workflow.pmservice.TransmissionType tp =
                        com.ca.workflow.pmservice.TransmissionType.fromString(
                                com.ca.workflow.pmservice.TransmissionType._MIME);
                //Image type is PNG.
                com.ca.workflow.pmservice.ImageType IT =
                        com.ca.workflow.pmservice.ImageType.fromString(
                                com.ca.workflow.pmservice.ImageType._PNG);
```

```
                service.getInstanceImage(sessid,instanceId, tp, IT);
                Object[] attach = ((PmServiceSoapBindingStub)service).getAttachments();
                if(attach.length == 0){
                        throw new Exception("Image attachment is missing");
                }
                org.apache.axis.attachments.AttachmentPart part =
                                (org.apache.axis.attachments.AttachmentPart)attach[0];
                DataHandler dh = part.getDataHandler();
                java.io.InputStream ips = dh.getInputStream();
                java.io.FileOutputStream fos = new java.io.FileOutputStream(imgFilePath);
                byte data[] = new byte[1024];
                int idataread = -1;
                while((idataread=ips.read(data)) != -1 ){
                        fos.write(data, 0, idataread);
                }
                fos.close();

        }
}
```

Glossary

ISBN number

ISBN is the abbreviation for International Standard Book Number, a unique, numerical, commercial book identifier.

SOAP

SOAP is an abbreviation for Simple Object Access Protocol, a simple XML-based protocol to let applications exchange information.

WSDL

WSDL is an abbreviation for Web Services Description Language, an XML language for describing Web services.

Index

L

O

P

R

S

T

U

V

W