CA Process Automation

Web Services API Reference

Release 04.2.00

CA Technologies Product References

This document references the following CA Technologies products:

- CA Catalyst for CA Service Desk Manager (CA Catalyst Connector for CA SDM)
- CA Client Automation (formerly CA IT Client Manager)
- CA Configuration Automation (formerly CA Cohesion® Application Configuration Manager)
- CA Configuration Management Database (CA CMDB)
- CA eHealth®
- CA Embedded Entitlements Manager (CA EEM)
- CA Infrastructure Insight (formerly Bundle: CA Spectrum IM & CA NetQoS Reporter Analyzer combined)
- CA NSM
- CA Process Automation (formerly CA IT Process Automation Manager)
- CA Service Catalog
- CA Service Desk Manager (CA SDM)
- CA Service Operations Insight (CA SOI) (formerly CA Spectrum® Service Assurance)
- CA SiteMinder®
- CA Workload Automation AE

Contact CA Technologies

Contact CA Support

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

- 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

Providing Feedback About Product Documentation

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

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: Web Services Introduction

Web services are application programming interfaces (APIs) that provide users with the ability to communicate with CA Process Automation from external sources.

CA Process Automation provides two methods to use Web services: REST and SOAP. This reference describes the various APIs that you can use to communicate using both methods.

Chapter 2: RESTful API Reference

REST is an alternative to SOAP-based Web services. Using Catalyst Process Automation Services, CA Process Automation can be accessed via headless operation through Catalyst RESTful services.

This section contains the following topics:

RESTful Web Services and CA Process Automation

REST is a design method that is based on HTTP. REST lets Web services identify and manipulate resources.

The implementation of RESTful services adheres to the following design principles:

- Use HTTP methods explicitly
- Be stateless
- Access resources such as URIs
- Support XML and JSON (JavaScript Object Notation)

The RESTful architectural style has achieved widespread adoption on the Web. CA Process Automation exposes RESTful services through Catalyst Process Automation Services, which RESTful client technology can consume. Designers can use any REST client to access CA Process Automation in headless mode, which allows configuration and control CA Process Automation without using the UI.

This chapter describes RESTful requests that a designer can send to Catalyst Process Automation Services to interact with CA Process Automation.

For more information, see the following documentation:

- The Catalyst Process Automation Services topics (see page 10) for information about how to run RESTful Web Services with the CA Process Automation operators.
- The *Content Administrator Guide* for information about how to deploy Catalyst Process Automation Services.
- The *Content Designer Reference* for more information about the Catalyst operators.

Catalyst Process Automation Services

The Catalyst container and Catalyst Process Automation Services are embedded in CA Process Automation. Communication is optimized among the Catalyst container, Catalyst Process Automation Services, and CA Process Automation.

Catalyst Process Automation Services include an embedded connector that exposes the UCF interfaces. Any UCF client application (including the CA Process Automation Catalyst operators) can use the UCF interfaces.

Any Catalyst client can use the CA Process Automation connector to perform the following actions:

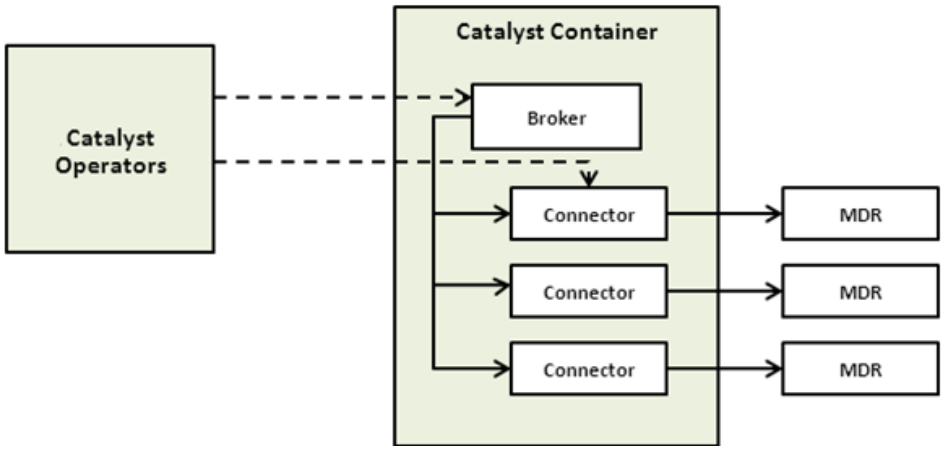
- Query the status of CA Process Automation processes
- Start, abort, suspend, and resume a CA Process Automation process
- Query and subscribe to alerts
- Query and submit start request forms
- Dequeue and end start requests
- Query, update, and respond to tasks (interaction requests)
- Import and export automation objects
- Receive lifecycle events for CA Process Automation processes
- Query and update datasets
- Query and update CA Process Automation modules

Important! Do not install other connectors in the Catalyst Process Automation container.

Usage

Any Catalyst client interface, including the Catalyst REST interface and the Catalyst operators that CA Process Automation includes, lets you access Catalyst Process Automation Services.

All Catalyst operators interact with Catalyst connectors, which interact with a management data repository (MDR). A Catalyst connector always resides in a Catalyst container. The Catalyst container has a broker (a node lookup service) that publishes the node configurations for each connector in the container.



All of the Catalyst operators require the Catalyst broker URL and the connector MdrProdInstance. When you enter a Catalyst broker URL in a Catalyst operator, the operator queries the broker for node configurations. The broker then populates the MdrProdInstance with a list of those node configurations.

After the Catalyst broker URL and the MdrProdInstance are defined in a Catalyst operator, you can specify the REST operations to run.

Communications

Catalyst Process Automation Services require secure Catalyst communications. The broker URL of Catalyst Process Automation Services is:

`https://<hostname>:7443/ucf/BrokerService`

The secure broker requires Catalyst credentials. If the secure broker is configured in the OasisConfig.properties "use.catalyst.claims.credentials" setting, it also requires CA Process Automation credentials. You can specify these credentials in the UCF Security section of any Catalyst operator. The credential values are entered in the Catalyst Security parameters for an operator. The claim names are Username and Password.

Catalyst Operators

To achieve the following goals, build CA Process Automation processes that use the Catalyst operators:

- Query the connector (Get operator)
- Invoke the connector operations (Execute operator)
- Subscribe to changes from the connector (SubscribeToChanges operator)
- Query CA Process Automation processes status
- Start, abort, suspend, and resume a CA Process Automation process
- Query and subscribe to alerts
- Query and submit start request forms
- Dequeue and end start requests
- Query, update, and respond to tasks (interaction requests)
- Import and export automation objects
- Receive lifecycle events about CA Process Automation processes
- Query and update datasets
- Query and update CA Process Automation modules

Note: The ucfpamconnector-descriptors.jar file contains the CA Process Automation connector descriptors.

Catalyst Security Parameters

All Catalyst operators include Catalyst Security parameters that support authentication at the Catalyst level and at the connector level.

After you get access to the Catalyst nodes, you can use claims to get connector-specific security information. See the Connector guide that is provided with the applicable Catalyst connector for information about connector-specific claims.

Administrators at the Domain level set the default security settings for the Catalyst operator category (which contains *all* of the Catalyst operators). These configurations are inherited. You can edit these settings for individual operators at the environment, Orchestrator, and agent levels.

See the *UI Reference* for information about security parameters at the operator category (or module) level. See the *Content Designer Reference* for information about security parameters for each operator.

Execute Parameters

Every Catalyst operator requires specific input parameters to execute successfully. See the *Content Designer Reference* for more information.

Features

Catalyst Process Automation Services support the following features.

USM Model Mapping

The extensible Unified Service Model (USM) schema models the relationships and policies that govern how the Catalyst Process Automation Services federate, correlate, and reconcile data and capabilities across CA Process Automation.

Catalyst Process Automation Services support the following types of USM model mapping:

USM Type and Property	**CA Process Automation Type and Property**
ITActivityTemplate	**Flowchart**
MdrProduct	"CA:00074"
MdrProdInstance	"CA:00074:01"
MdrElementID	ReferencePath + ReferenceName
UrlParams	"Tear off" URL UrlParams contains the CA Process Automation tear off URL. Paste the URL in the browser and log in to CA Process Automation. The user interface displays an Automation Object.
CreationTimestamp	CreationDate
LastModTimestamp	ModifiedDate
LastModUserName	EditedBy
ActivityTypes	"Workflow"
DefinitionName	ReferencePath + ReferenceName
DefinitionVersion	Version
ITActivity	**Workflow**
MdrProduct	"CA:00074"

MdrProdInstance	"CA:00074:01"
MdrElementID	Instance
UrlParams	"Tear off" URL UrlParams contains the CA Process Automation tear off URL. Paste the URL in the browser and log in to CA Process Automation. The user interface displays an Automation Object.
CreationTimestamp	CreationDate
LastModTimestamp	ModifiedDate
LastModUserName	EditedBy
ActivityID	ROID
RuntimeName	Instance
RuntimeDiscriminator	**ROID**
ActivityTypes	"Workflow"
StateDescription	RunState
ActivityState:	RunState:
"Finished-Completed"	"Completed"
"Finished-Completed"	"completedResponse"
"Finished-Abandoned"	"Aborted"
"Finished-Failed"	"Failed"
"Normal-Running"	"Running"
"Normal-Waiting"	"Idle"
"Obstructed"	"Suspended"
"Obstructed"	"BreakPointSuspended"
"Normal-Waiting"	"Blocked"
"AwaitingScheduling"	"Queued"
BinaryRelationship	**Workflow**
MdrProduct	"CA:00074"
MdrProdInstance	"CA:00074:01"
MdrElementID	Instance (of Workflow)

SourceMdrProduct	"CA:00074"
SourceMdrProdInstance	"CA:00074:01"
SourceMdrElementID	Instance (of Workflow)
TargetMdrProduct	"CA:00074"
TargetMdrProdInstance	"CA:00074:01"
TargetMdrElementID	ReferencePath + ReferenceName (of Flowchart)
Semantic	"IsInstanceOf"
Alert	**Task or Workflow**
MdrProduct	"CA:00074"
MdrProdInstance	"CA:00074:01"
MdrElementID	Instance (Workflow) or "Task:" + TaskID (Task)
OccurrenceTimestamp	ModifiedDate(Workflow) or Started(Task)
AlertType	"Risk" (Workflow) or "Informational" (Task)
Severity	"Major" (Workflow) or "Informational" (Task)
AlertedMdrProduct	"CA:00074"
AlertedMdrProdInstance	"CA:00074:01"
AlertedMdrElementID	Instance
Summary	Instance + RunState (Workflow) or TaskID + RunState + PromptPath (Task)
Message	EventDescription (Workflow) or PromptTitle + PromptDescription (Task)
UrlParams	"tear off" URL UrlParams contains the CA Process Automation tear off URL. Paste the URL in the browser and log in to CA Process Automation. The user interface displays an Automation Object.

Querying Capabilities

The Catalyst Process Automation Services can use the get() operation on the supported USM types. Queries support the "id" and "updatedAfter" properties of the basic filter. A "get" for alerts returns tasks or pending interaction requests.

Custom Operations

Catalyst Process Automation Services support custom operations. Custom operations are available for:

- Processes
- Start requests
- Tasks
- Import/Export of content
- Datasets
- Module configuration

Note: Custom operations require secure access. If the "use.catalyst.claims.credentials" value is true, then claims are also required for CA Process Automation credentials.

In the Execute operator (inside the Catalyst module), select a custom operation from the Operation drop-down list.

Process Operations

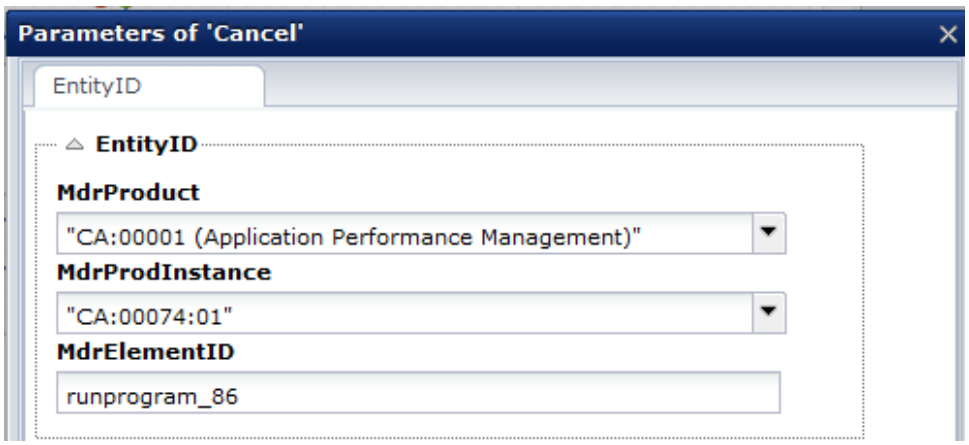
Process operations let you act on a process:

Start a process

Use the MdrElementID of an ITActivityTemplate.

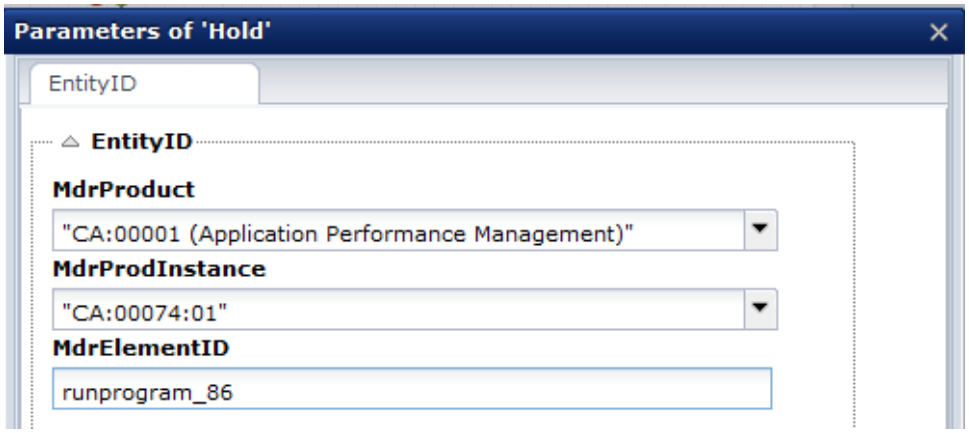
Cancel (abort) a process

Use the MdrElementID of an ITActivity.



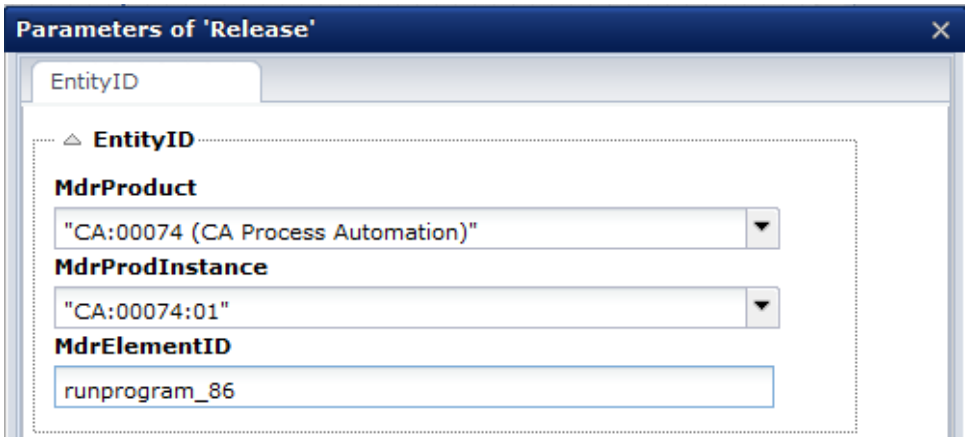
Hold (suspend) a process

Use the MdrElementID of an ITActivity.



Release (resume) a process

Use the MdrElementID of an ITActivity.



Start Request Operations

Start request operations let you conduct the following actions on start request forms:

- QueryStartRequestForms
- SubmitStartRequestForm
- QueryStartRequests
- DequeueStartRequest
- AbortStartRequest

Examples

The following topics explain how to use start request operations with the Catalyst Execute operator.

QueryStartRequestForms

To query start request forms, use the QueryStaonrtRequestForms operation.

Follow these steps:

1. Create a process.
2. Drag the Execute operator to the Designer.
3. Double-click the Execute operator to open the Properties palette.
4. Enter the required Catalyst Security and the required Execute parameters.

   Catalyst Security

   Username: *(Catalyst user name)*

   Password: *(Catalyst password)*

   Claims:

   ClaimName: Username

   ClaimValue: pamadmin

   PasswordClaims:

   ClaimName: Password

   ClaimValue: pamadmin

   Execute

   CatalystBrokerURL: https://*hostname*:7443/ucf/BrokerService

   MdrProduct: "CA:00074 (CA Process Automation)"

   MdrProdInstance: "CA:00074:01"

   Operation Category: "CA Process Automation"

   Operation: QueryStartRequestForms

5. Create a recursive filter that queries start request forms in the Library.The filter can restrict the search by path, recursively, and by keywords (tags).

   a. In the Execute Properties, click Parameters.

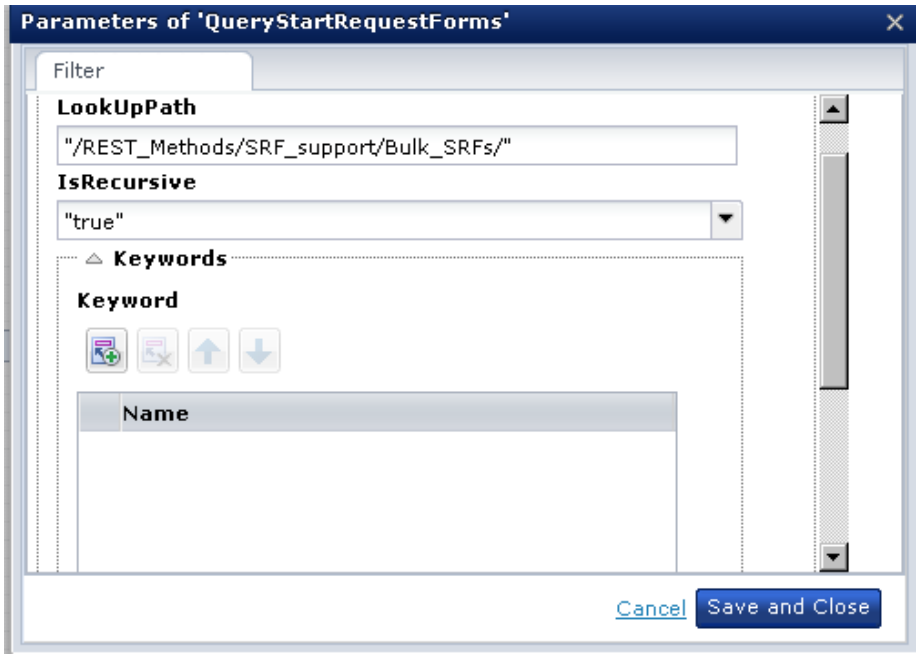
   b. On the Filter tab, enter the following parameters:

      LookUpPath: Defines the path to query for start request forms. In this example, we are querying for start request forms in the "Bulk_SRFs" parent folder.

      IsRecursive: true

      Keywords: Defines one or more keywords (such as run, queued, and complete) with which to query start request forms.

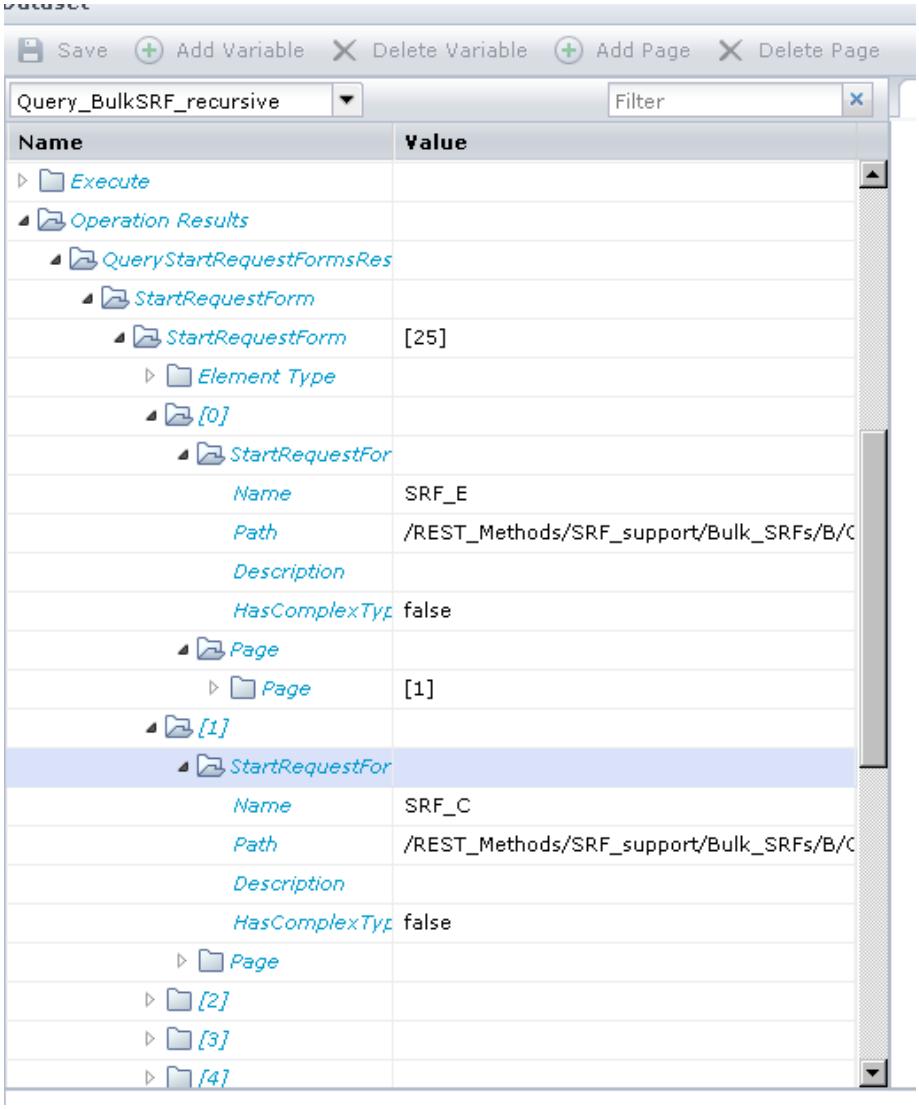
      You can also conduct a query without keywords, as in the following illustration:



   c. Click Save and Close.

6. Run the process.
7. Review the output.

   For example, in the following illustration there are 25 start request forms (with variables) from the provided LookUpPath:



SubmitStartRequestForm

To submit a start request form, use the SubmitStartRequestForm operation.

Follow these steps:

1. Create a process.
2. Drag the Execute operator to the Designer.

3. Double-click the Execute operator to open the Properties palette.
4. Enter the required Catalyst Security and Execute parameters.

Catalyst Security

Username: *(Catalyst user name)*

Password: *(Catalyst password)*

Claims:

ClaimName: Username

ClaimValue: pamadmin

PasswordClaims:

ClaimName: Password

ClaimValue: pamadmin

Execute

CatalystBrokerURL: https://*hostname*:7443/ucf/BrokerService

MdrProduct: "CA:00074 (CA Process Automation)"

MdrProdInstance: "CA:00074:01"

Operation Category: "CA Process Automation"

Operation: SubmitStartRequestForm

5. Submit a start request form to a queued state.
   a. In the Execute Properties, click Parameters.
   b. On the SubmitStartRequestFormRequest tab, enter the following parameters:

      Name:

      "SRF_Queue"

      Path:

      The absolute path of the start request form (you can get the path from the properties of a start request form). For example:
      SRF"/REST_Methods/SRF_support/SRFs_test/".

   c. Enter arguments in the Arguments tab. For example:

      Name:

      Var_0

      Value:

      from_proc

      and

      Name:

      Var_1

      Value:

      from_prc2

   d. Enter date and time parameters in the Options tab. For example:

      StartDate:

      "2012-08-31" (yyyy-MM-dd)

      StartTime:

      "12:25" (HH:mm)

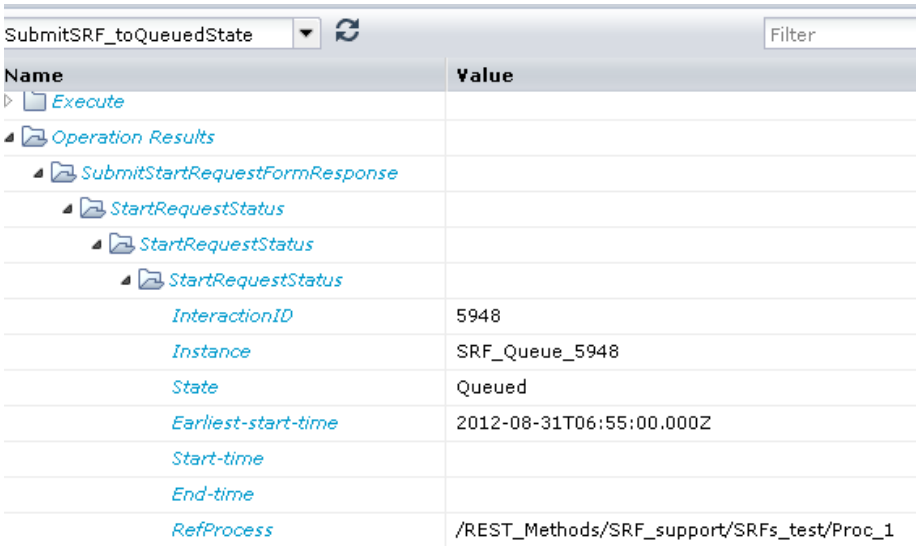
      Priority:

      No value

      Note: If you do not set StartDate and StartTime, CA Process Automation schedules the start request form immediately.

6. Run the process.
7. Review the output.

   When the operator runs, CA Process Automation schedules the specified start request form in the server. As the following illustration shows, the output parameters include results such as the start request form instancesID, State, and associated process details.



QueryStartRequests

The QueryStartRequests operation lets you query a start request form instance for the start request Status and the associated Process instance ID.

Follow these steps:

1. Create a process.
2. Drag the Execute operator to the Designer.
3. Double-click the Execute operator to open the Properties palette.

4. Enter the required parameters.

   Catalyst Security

   Username: *(Catalyst user name)*

   Password: *(Catalyst password)*

   Claims:

   ClaimName: Username

   ClaimValue: pamadmin

   PasswordClaims:

   ClaimName: Password

   ClaimValue: pamadmin

   Execute

   CatalystBrokerURL: https://*hostname*:7443/ucf/BrokerService

   MdrProduct: "CA:00074 (CA Process Automation)"

   MdrProdInstance: "CA:00074:01"

   Operation Category: "CA Process Automation"

   Operation: QueryStartRequests

5. Identify the start request form instance that is in the Queued state.

   a. In the Execute Properties, click Parameters.

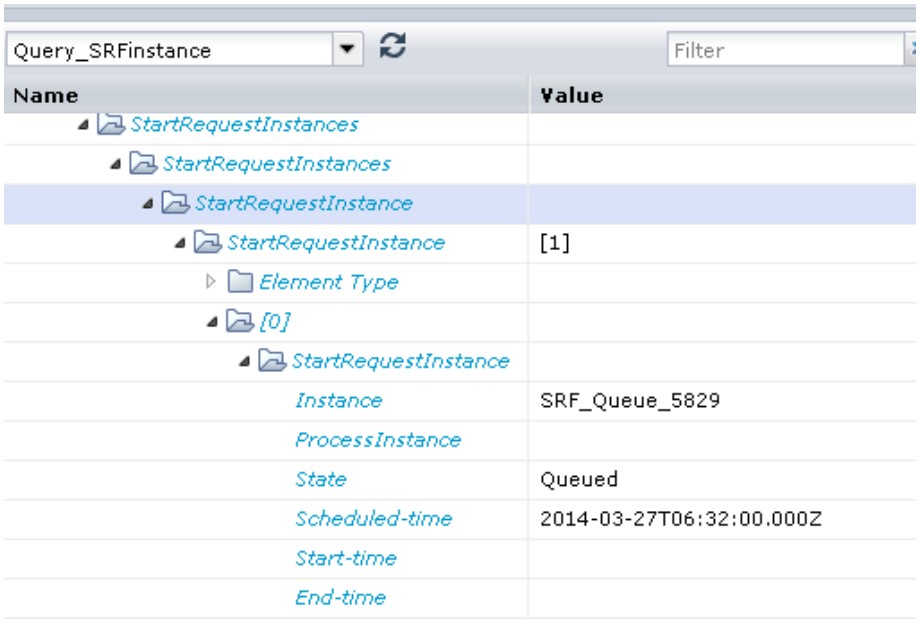
   b. On the QueryStartRequestsRequest tab, enter the following values:

   Instance: "SRF_Queue_*x*" (where *x* is the instance ID of the start request form that you want to query). For example, "SRF_Queue_5829".

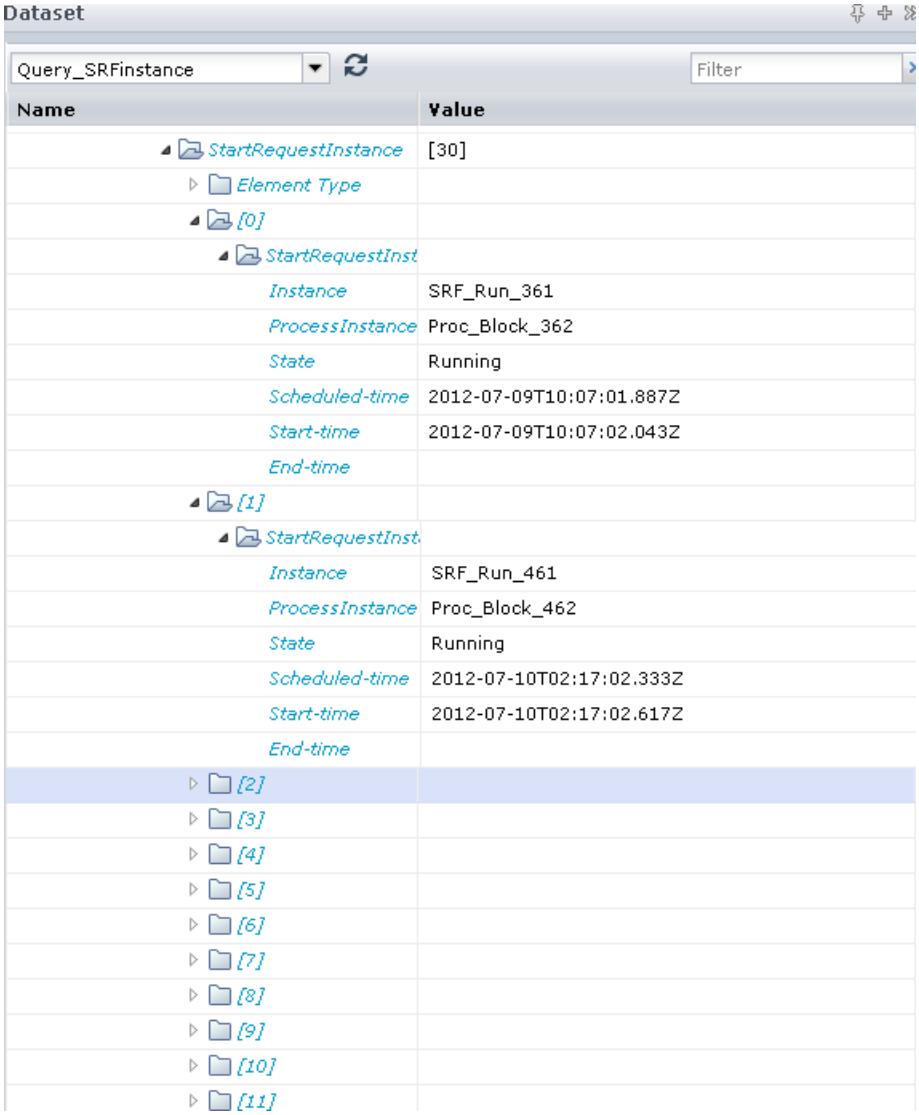
   IsArchived: "false"

6. Run the process.
7. Review the output.

   In the following illustration, the output parameters display the start request form in the queued state.



In the following illustration, the output parameters show a list of all the archived start request form instances. To get archived results, do not include the instance name in the input parameters, but provide true or false for the IsArchived parameter.



DequeueStartRequest

The DequeueStartRequest operation removes a start request form instance from the queue.

Follow these steps:

1. Create a process.
2. Drag the Execute operator to the Designer.

3. Double-click the Execute operator to open the Properties palette.
4. Enter the required parameters.

   Catalyst Security

   Username: *(Catalyst user name)*

   Password: *(Catalyst password)*

   Claims:

   ClaimName: Username

   ClaimValue: pamadmin

   PasswordClaims:

   ClaimName: Password

   ClaimValue: pamadmin

   Execute

   CatalystBrokerURL: https://*hostname*:7443/ucf/BrokerService

   MdrProduct: "CA:00074 (CA Process Automation)"

   MdrProdInstance: "CA:00074:01"

   Operation Category: "CA Process Automation"

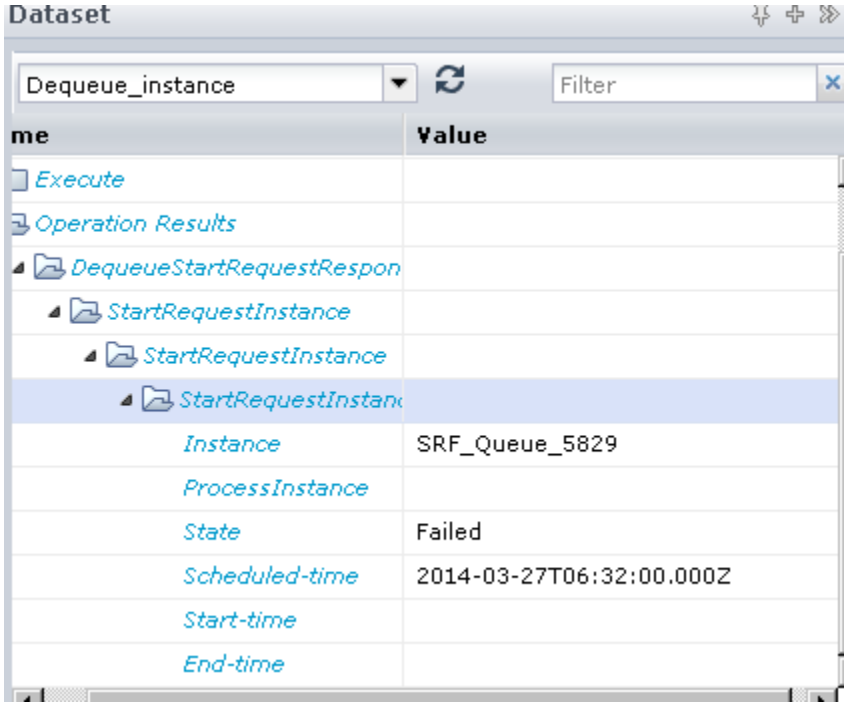
   Operation: DequeueStartRequest

5. Dequeue the start request form instance that is in the Queued state.
   a. In the Execute Properties, click Parameters.
   b. In the DequeuetStartRequestRequest tab, enter the Instance value "SRF_Queue_x" (where *x* is the instance ID of the request form to dequeue). For example, "SRF_Queue_5829."

6. Run the process.
7. Review the output.

   In the following illustration, the output parameters display the dequeued start request form. The State returns as Failed because the start request form instance was dequeued.

   Note: You can only dequeue the queued start request form instance.



AbortStartRequest

To end a start request form instance, use the AbortStartRequest operation.

Follow these steps:

1. Create a process.
2. Drag the Execute operator to the Designer.
3. Double-click the Execute operator to open the Properties palette.

4. Enter the required parameters.

   Catalyst Security

   Username: *(Catalyst user name)*

   Password: *(Catalyst password)*

   Claims:

   ClaimName: Username

   ClaimValue: pamadmin

   PasswordClaims:

   ClaimName: Password

   ClaimValue: pamadmin

   Execute

   CatalystBrokerURL: https://*hostname*:7443/ucf/BrokerService

   MdrProduct: "CA:00074 (CA Process Automation)"

   MdrProdInstance: "CA:00074:01"

   Operation Category: "CA Process Automation"

   Operation: AbortStartRequest

5. End the running start request form instance:

   a. In the Execute Properties, click Parameters.

   b. In the AbortStartRequestRequest tab, enter the following value:

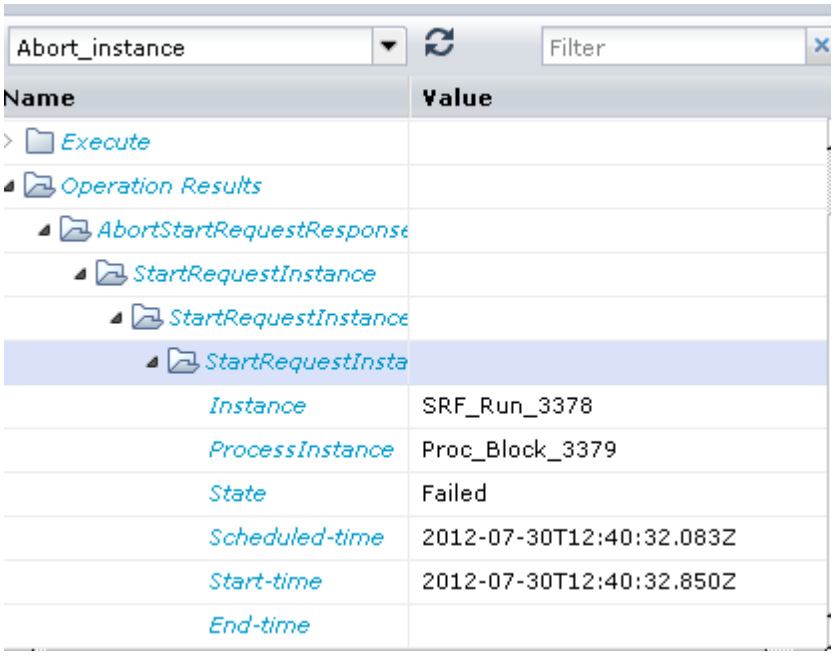
   Instance:

   "SRF_Run_*x*" (where *x* is the instance ID of the start request form to dequeue). For example, "SRF_Run_3378."

6. Run the process.
7. Review the output.

   In the following illustration, the output parameters display the current state of the start request form as Failed because the Running state was ended.

   Note: You can only end the running start request form instance.



Task Operations

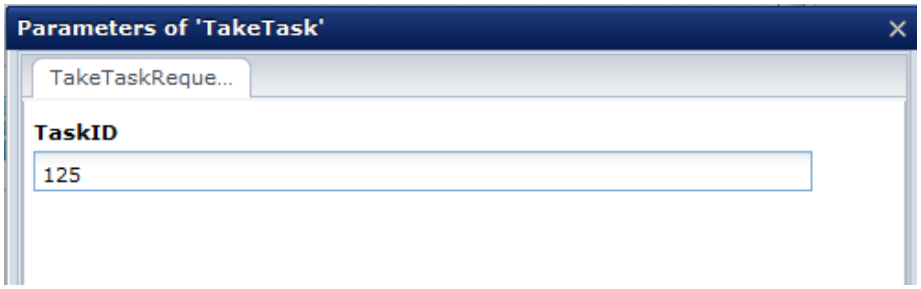
Task operations let you take the following actions on tasks:

QueryTasks

Query tasks (pending interaction requests).

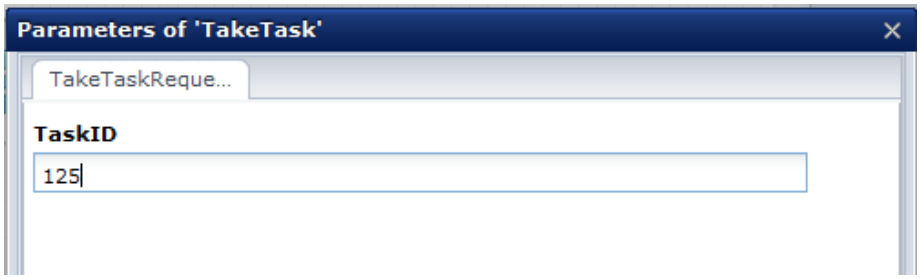
ReplyTask

Reply to a task with parameters and approval.



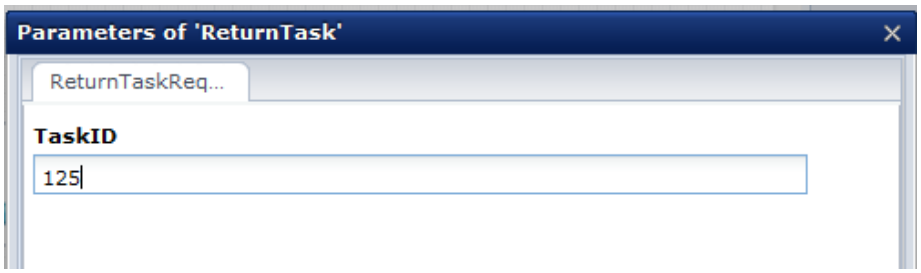
TakeTask

Take a task.



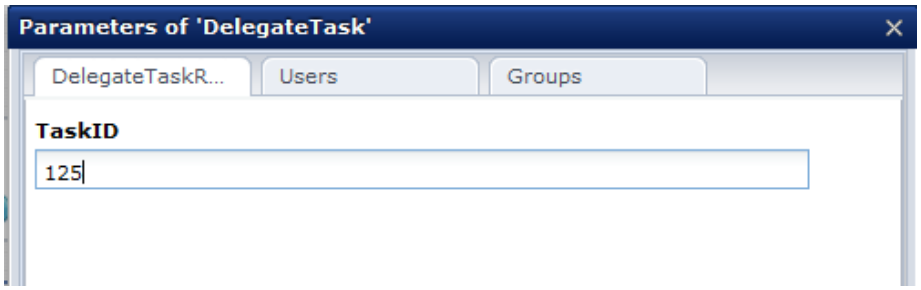
ReturnTask

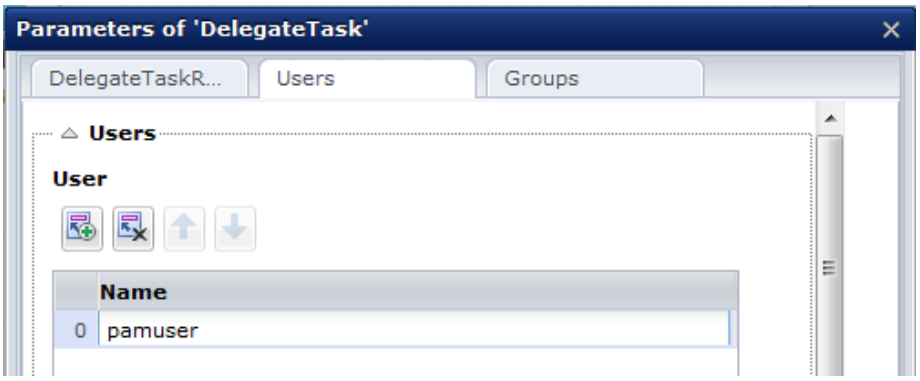
Return a task.



DelegateTask

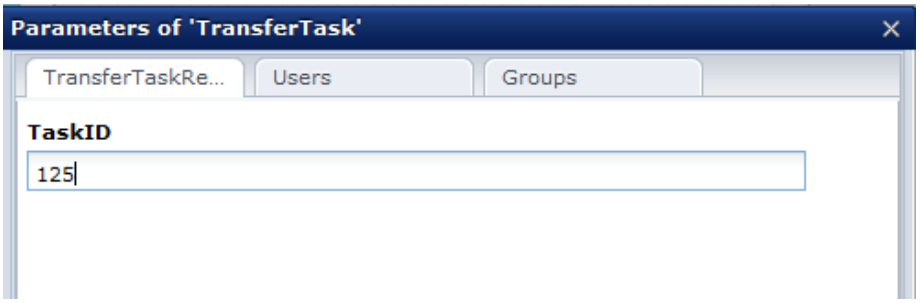
Delegate a task to one or more users or groups.

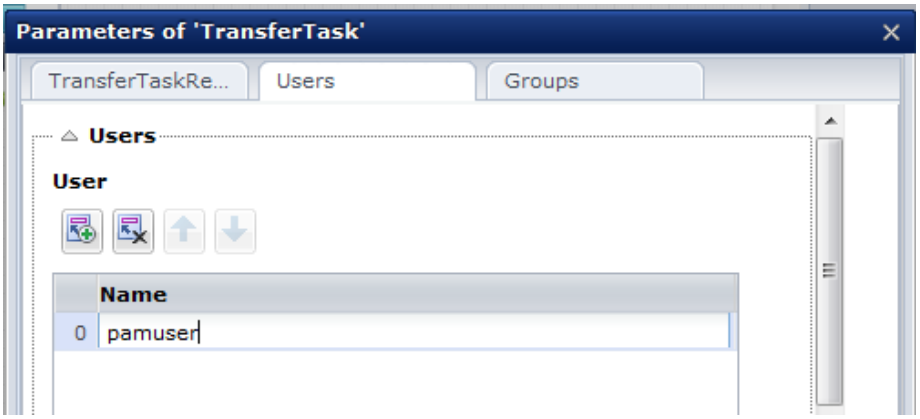




TransferTask

Transfer (that is, assign) a task to one or more users or groups.





Content Operations

Content operations let you export content from one CA Process Automation environment and import it into another CA Process Automation environment.

Import

Import operation lets you import a local file of CA Process Automation objects. During an import, you can set the <OverwriteAction> to one of the following options:

- Import
- DoNotImport
- ImportAndReplace

Using the Execute operator, select the Import operation, then provide the values.

Example

RESTful services let you import content from a server location.

Follow these steps:

1. Create a process.
2. Drag the Execute operator to the Designer.
3. Double-click the Execute operator to open the Properties palette.
4. Enter the required parameters.

   Catalyst Security

   Username: (*Catalyst user name)*

   Password: (*Catalyst password*)

   Claims

   ClaimName: Username

   ClaimValue: pamadmin

   PasswordClaims

   ClaimName: Password

   ClaimValue: pamadmin

   Execute

   CatalystBrokerURL: "https//hostname:7443/ucf/BrokerService"

   MdrProduct: "CA:00074 (CA Process Automation)"

   MdrProdInstance: "CA:00074:01"

   Operation Category: "CA Process Automation"

   Operation: Import

5. Determine which of the following options to use when importing content, then complete the Server Location fields:

   - Import from the server location
   - Import from the HTTP URL

   Server Location

   a. In the Execute Properties, click Parameters.

   b. On the ImportRequest tab, complete the following fields:

      ImportLocation: Defines the Library location to which to import the XML file. If the specified folder does not exist, CA Process Automation creates it. For example, "/test-import/".

      SourceLocation: Define the source folder in one of the following ways:

      - Define the server location from which to import the content. For example, "c:\\export_import_test\\export_test.xml".
      - Define the HTTP URL for the location from which to import the content. For example, "http://servername:8080/c2orepository/test/export_test.xml".

      OverwriteAction: Enter one of the following options to import a file of the same name that exists in the specified location:

      - **Import:** Increase the version of the imported object if an object of the same exists in the specified import folder. Overrides the release version if the same release version exists.
      - **DoNotImport:** Skip the import operation if an object of the same name exists in the specified import folder.
      - **ImportAndReplace:** Replace the object of the same name in the specified import folder. It deletes the previous versions of the object.

      SetCurrent: Select "true" to set the imported version as current version; otherwise, select "false".

      MakeAvailable: Select "true" to make the content available in the Library. Select "false" to make the content unavailable.

6. Run the process.

7. Review the output.

   The following illustration shows a successful import message:

Export

Export operations let you export a local file of CA Process Automation objects into a local file. The filter can use the object type to restrict which objects are exported.

Using the Execute operator, select the Export operation, then provide the values.

Example

To export content with a relative path, use RESTful services.

Follow these steps:

1. Create a new process.
2. Drag the Execute operator to the Designer.
3. Double-click the Execute operator to open the Properties palette.
4. Enter the required parameters.

   Catalyst Security

   Username: *(Catalyst user name)*

   Password: *(Catalyst password)*

   Claims:

   ClaimName: Username

   ClaimValue: pamadmin

   PasswordClaims:

   ClaimName: Password

   ClaimValue: pamadmin

   Execute

   CatalystBrokerURL: "https//hostname:7443/ucf/BrokerService"

   MdrProduct: "CA:00074 (CA Process Automation)"

   MdrProdInstance: "CA:00074:01"

   Operation Category: "CA Process Automation"

   Operation: Export

5. Export the content with a relative path.

   a. In the Execute Properties, click Parameters.

   b. In the ExportRequest tab, enter the following values:

      FolderName: Absolute path of the folder (or) content. For example, "/GetOSInfo".

      ExportAsContentPackage: Export folder as Content Package with absolute or a relative path. Takes true or false value. By default, the value is set to false.

      IsAbsolute: Select "false" to export the content with a relative path. If you select "true", content will be imported with the absolute path.

   c. In the ExportFileName tab, enter the following values:

      ExportLocation: Server location to export the .xml file. For example: "c:\\export_import_test\\export_test.xml".

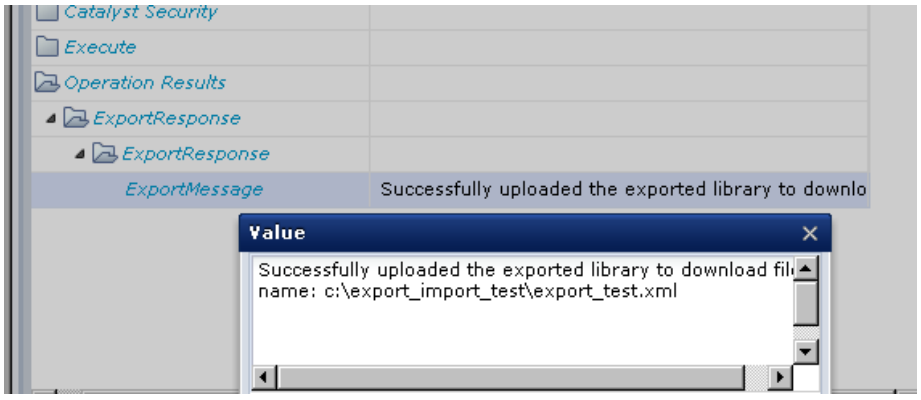
      OverwriteFile: Select whether to overwrite the existing file in the given location.

   d. To export all the content to include any type of object, do not provide object types. When you want to export specific object types from the given location, provide types like "Process", "Dataset".

6. Run the process.

7. Review the output.

   In the following illustration, a successful export message displays with exported location.



Dataset Operations

The dataset operations let you query and update global named datasets.

Examples

The following topics explain how to use dataset operations with the Catalyst Execute operator.

QueryDatasetNames

Use the QueryDatasetNames operation to search the specified path for dataset names.

Follow these steps::

1. Create a new process.
2. Drag the Execute operator to the Designer.
3. Double-click the Execute operator to open the Properties palette.
4. Enter the required parameters.

   Catalyst Security

   Username: *(Catalyst user name)*

   Password: *(Catalyst password)*

   Claims:

   ClaimName: Username

   ClaimValue: pamadmin

   PasswordClaims

   ClaimName: Password

   ClaimValue: pamadmin

Execute

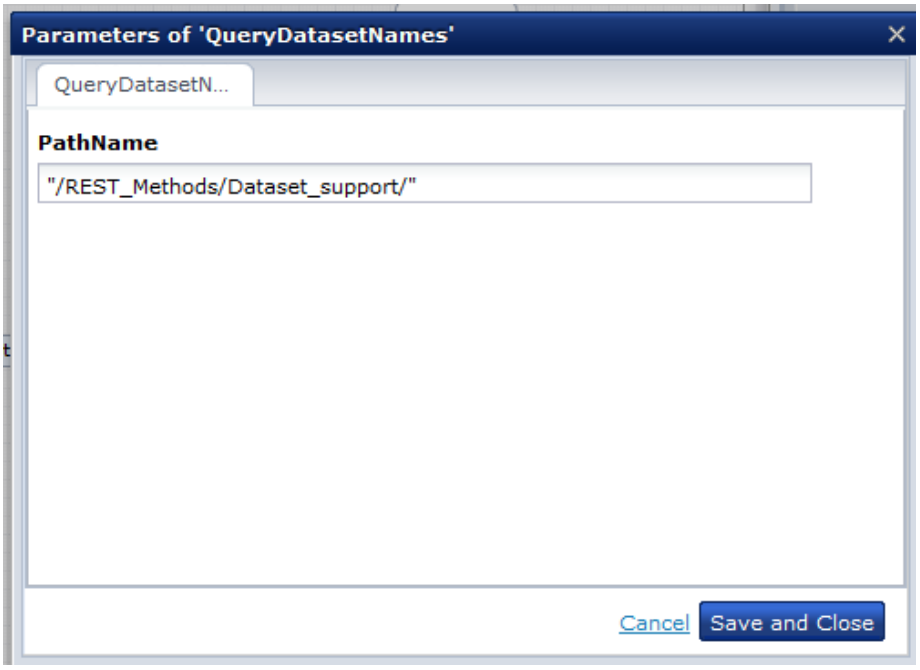
CatalystBrokerURL: https://*hostname*:7443/ucf/BrokerService

MdrProduct: "CA:00074 (CA Process Automation)"

MdrProdInstance: "CA:00074:01"

Operation Category: "CA Process Automation"

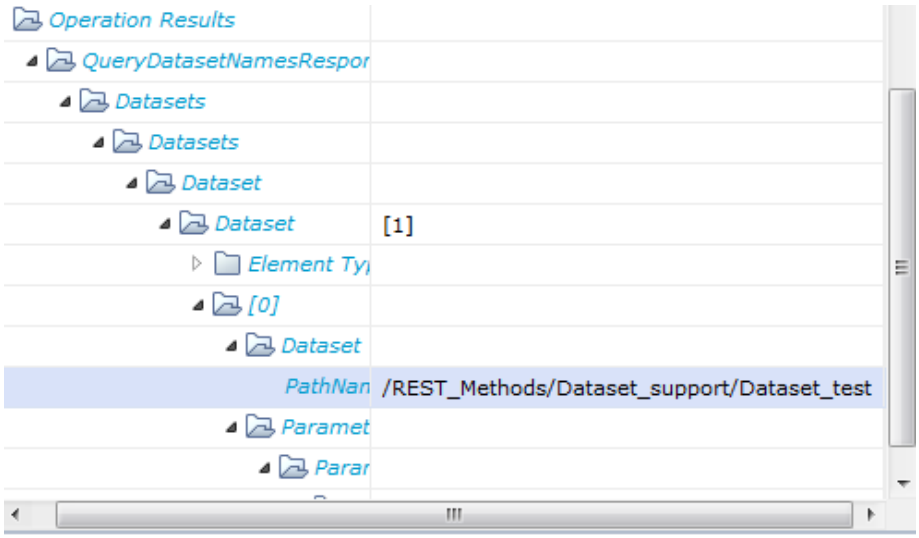
Operation: QueryDatasetNames



5. Run the process.
6. Review the output.

   This illustration shows the dataset name with the absolute path available in the following Library path:

   /REST_Methods/Dataset_support/Dataset_test



QueryDatasetParameters

Use the QueryDatasetParameters operation to search the specified path for dataset parameters.

Follow these steps::

1. Create a new process.
2. Drag the Execute operator to the Designer.
3. Double-click the Execute operator to open the Properties palette.
4. Enter the required parameters.

   Catalyst Security

   Username: *(Catalyst user name)*

   Password: *(Catalyst password)*

   Claims:

   ClaimName: Username

   ClaimValue: pamadmin

   PasswordClaims

   ClaimName: Password

   ClaimValue: pamadmin

Execute

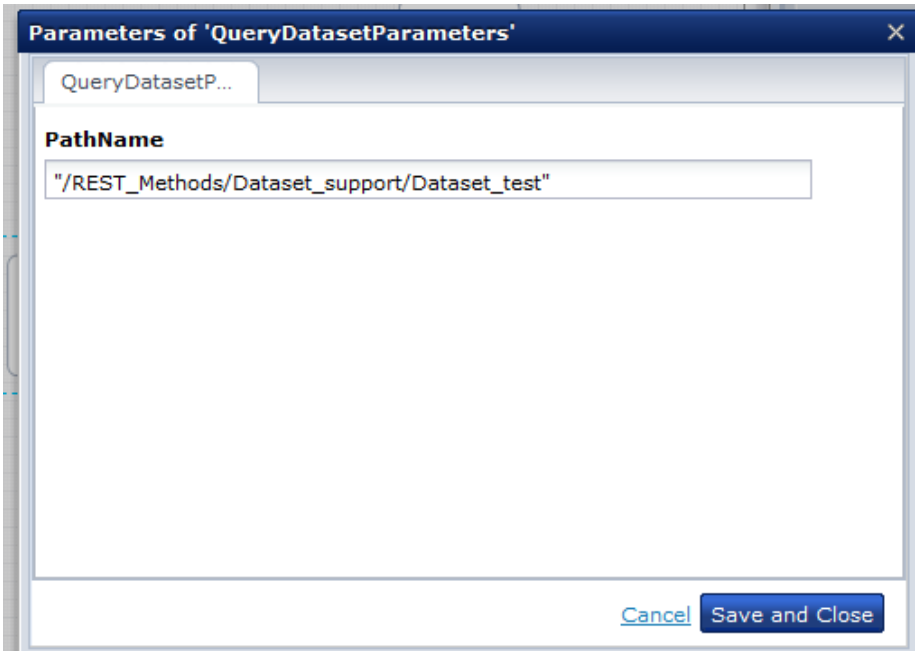
CatalystBrokerURL: https://*hostname*:7443/ucf/BrokerService

MdrProduct: "CA:00074 (CA Process Automation)"

MdrProdInstance: "CA:00074:01"

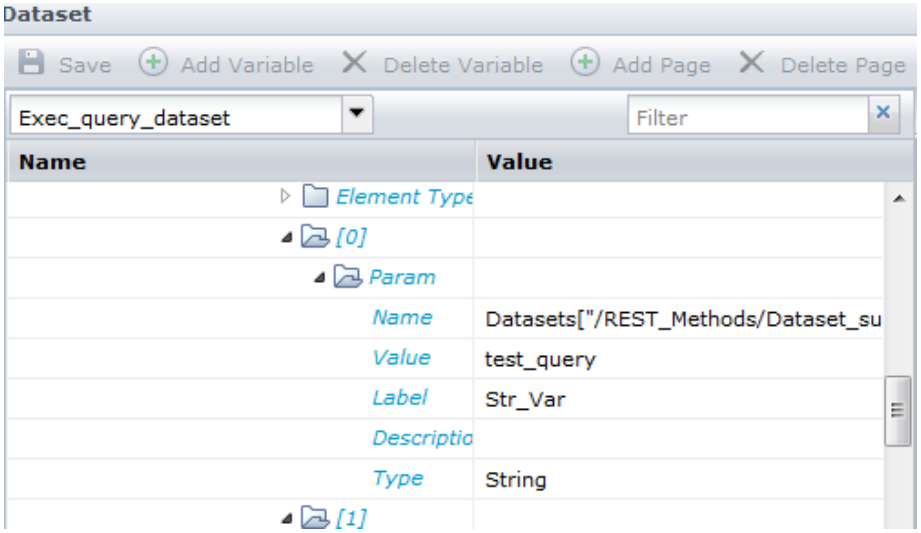
Operation Category: "CA Process Automation"

Operation: QueryDatasetParameters



5. Run the process.
6. Review the output.

   This illustration shows the dataset name with the Value, Type:



UpdateDatasetParameters

Use the UpdateDatasetParameters operation to update the parameters of a dataset.

Note: Before updating a global dataset, make sure that the dataset has been checked in at least once.

Follow these steps::

1. Create a new process.
2. Drag the Execute operator to the Designer.
3. Double-click the Execute operator to open the Properties palette.

4. Enter the required parameters.

Catalyst Security

Username: *(Catalyst user name)*

Password: *(Catalyst password)*

Claims:

ClaimName: Username

ClaimValue: pamadmin

PasswordClaims

ClaimName: Password

ClaimValue: pamadmin

Execute

CatalystBrokerURL: https://*hostname*:7443/ucf/BrokerService

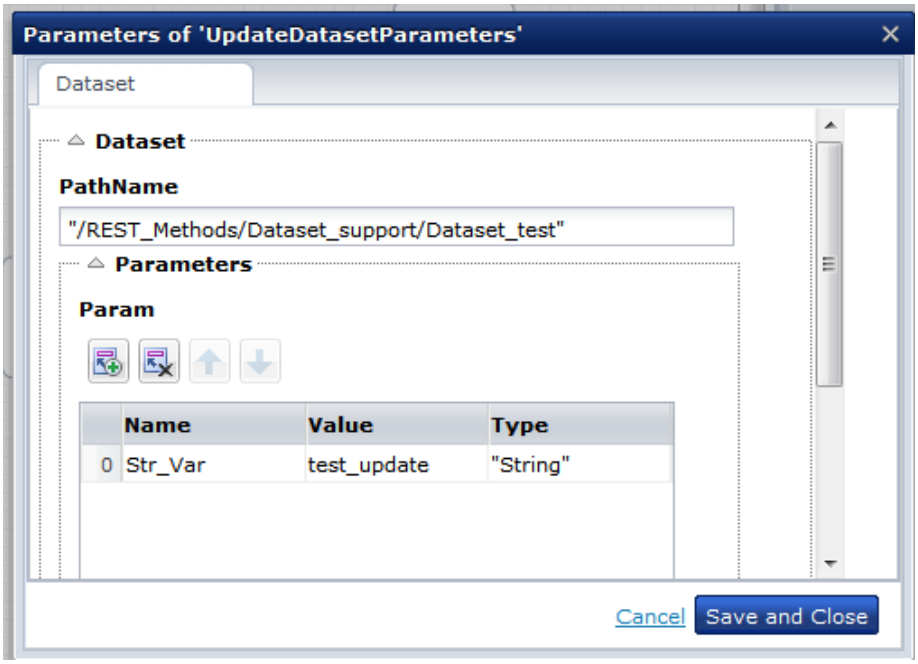
MdrProduct: "CA:00074 (CA Process Automation)"

MdrProdInstance: "CA:00074:01"

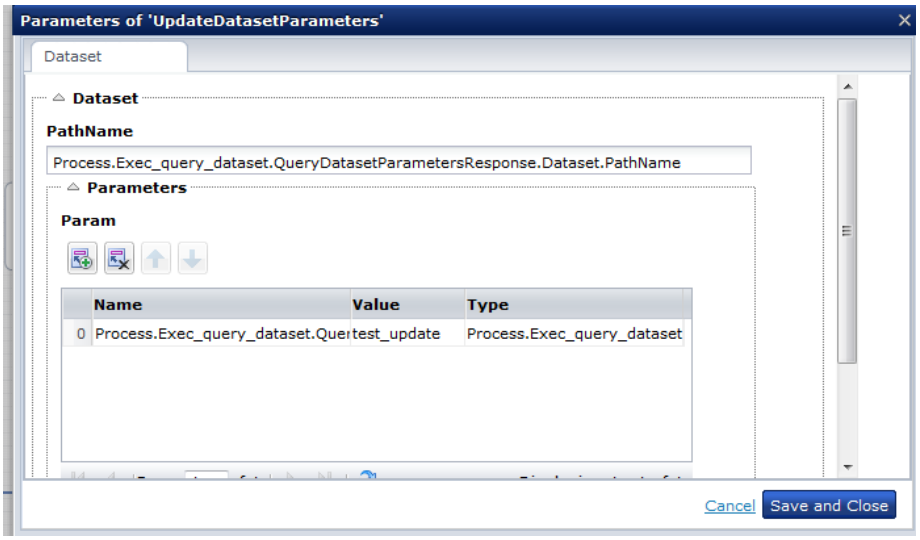
Operation Category: "CA Process Automation"

Operation: UpdateDatasetParameters

This example shows the dataset parameter of Type "string" with string value:

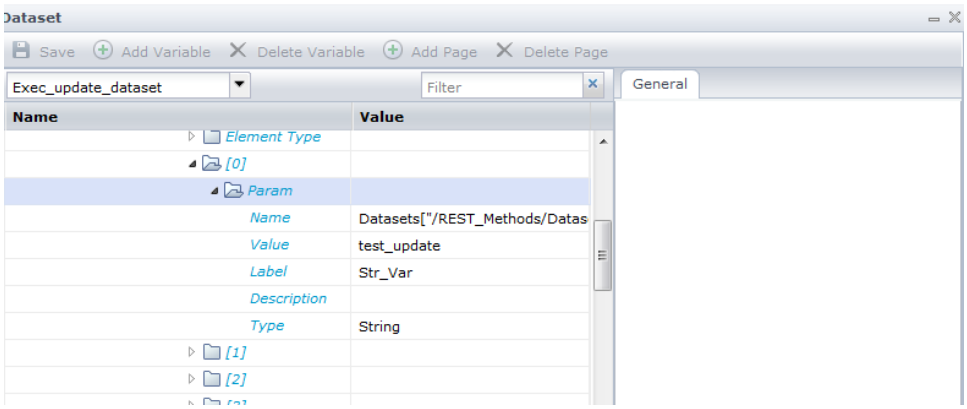


This example shows how to update the Dataset Parameter using an expression:



5. Run the process.
6. Review the output.

   This illustration shows how the dataset parameter of Type "String" is updated with the given value "test_update".



QueryProcessDataset

To return all of the parameters that a dataset contains, use the QueryDatasetParameters operation.

Follow these steps:

1. Create a process.
2. Drag the Execute operator to the Designer.

3. Double-click the Execute operator to open the Properties palette.
4. Enter the required parameters.

Catalyst Security

Username: *(Catalyst user name)*

Password: *(Catalyst password)*

Claims:

ClaimName: Username

ClaimValue: pamadmin

PasswordClaims

ClaimName: Password

ClaimValue: pamadmin

Execute

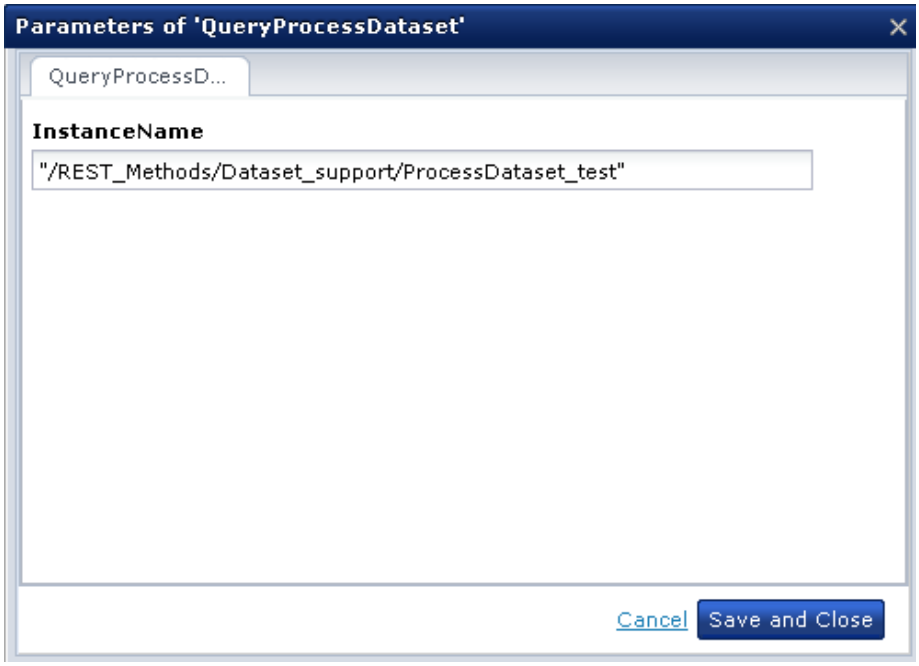
CatalystBrokerURL: https://*hostname*:7443/ucf/BrokerService

MdrProduct: "CA:00074 (CA Process Automation)"

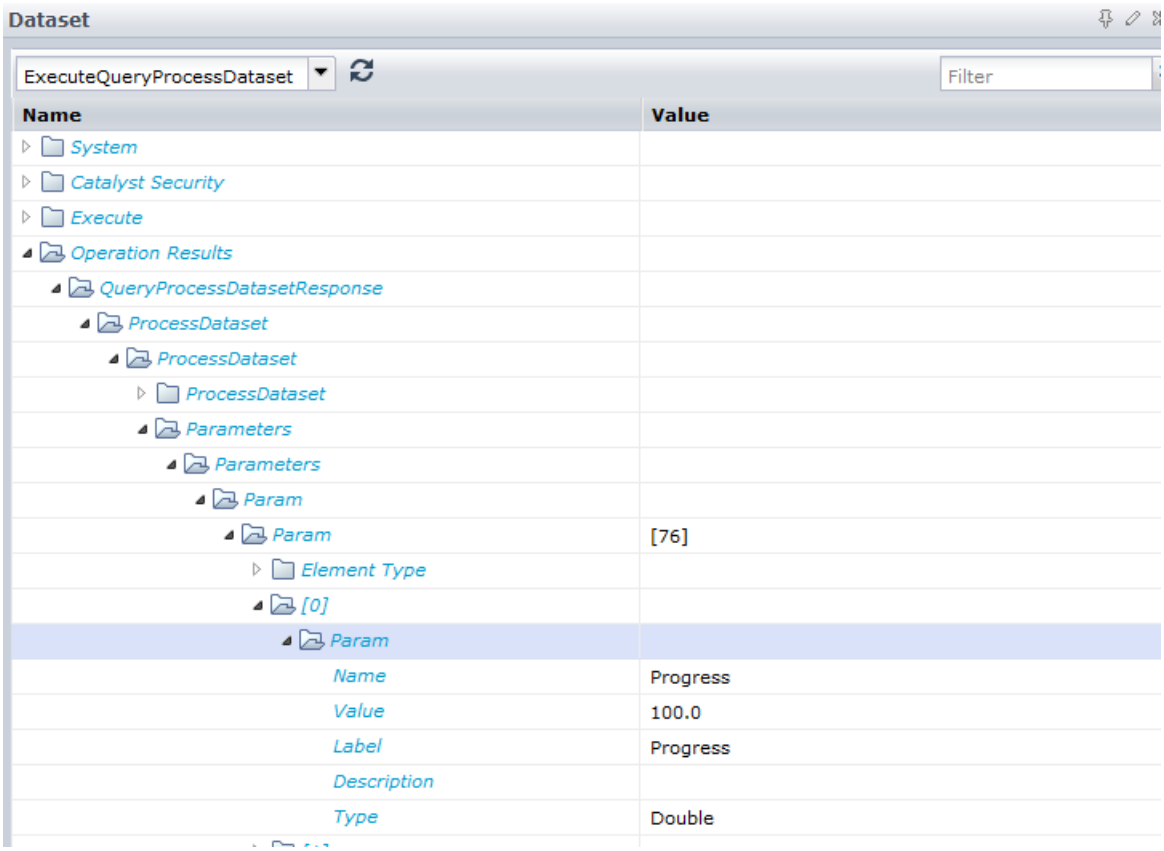
MdrProdInstance: "CA:00074:01"

Operation Category: "CA Process Automation"

Operation: QueryProcessDataset



5. Run the process.
6. Review the output.



Module Configuration Operations

Module configuration operations let you act on operator categories (or modules):

QueryModuleConfigs

Queries for modules.

(Optional) The <ModuleType> attribute specifies whether the standard module configuration or custom module group configuration is returned. By default, the QueryModuleConfigs method returns all the available modules. The <ModuleType> attribute takes the following values:

All

Returns the standard modules and the custom modules configuration.

Standard

Returns only standard modules configuration.

Custom

Returns only custom modules configuration.

QueryModuleConfigProperties

Searches for the available modules and the custom module properties.

UpdateModuleConfigProperties

Updates the properties of the available modules and custom modules.

Event Subscriptions

Catalyst Process Automation Services support the following event subscriptions:

- "Created" event when an ITActivity (process) starts.
- "Modified" event when an ITActivity (process) changes (that is, when the ActivityState changes).
- Alert "Created" event when an ITActivity (process) fails or is aborted.

Note: The "Deleted" event is not applicable.

Catalyst REST

Catalyst RESTful services let you access Catalyst Process Automation Services. To discover the Catalyst RESTful interface resources, start at the base URL from any browser and use HTTP GET requests to traverse the hyperlinks.

The base URL of the Catalyst RESTful interface is:

`http://<hostname>:7000/node/rest/`

The base URL displays the URLs of the Catalyst container modules. The Catalyst container hosts modules for the Catalyst broker and Catalyst Process Automation Services.

The status of the modules can be displayed at:

`http://<hostname>:7000/node/rest/broker/Entity`

The module for Catalyst Process Automation Services is "CA:00074_CA:00074:01":

`http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/`

The connector descriptors are accessed through the REST API metadata interface, which is:

`http://<hostname>:7000/node/rest/CA:00074_CA:00074:01;metadata=descriptor`

For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<conndesc:descriptor
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:conndesc="http://www.ca.com/connex/conndesc"
    xsi:schemaLocation="http://www.ca.com/connex/conndesc
conn_desc.xsd"
    xmlns:usm="http://www.ca.com/usm"
    xmlns:xi="http://www.w3.org/2001/XInclude"
    xmlns:usm-core="http://ns.ca.com/2009/07/usm-core"
    xmlns:pam-ops="http://ns.ca.com/2011/09/pam-ops"
    xmlns:pam="http://www.ca.com/pam">


       <ID>com.ca.c2o.mdr.UCFPAMConnector</ID>
       <version>4.0</version>
       <category>CA Process Automation</category>

       <configurationDesc xsi:type="usm:KeywordValuePairs">
       </configurationDesc>

       <connectorProperties>
              <property name = "BaseURI" value="http://pam-uri"/>
       </connectorProperties>


  <!-- ****************************** -->
  <!-- Process operations...          -->
  <!-- ****************************** -->

  <!-- Start a Process -->
  <customOp ID="Start" name="Start" synchronous="true">
    <inputType xsi:type="pam-ops:StartRequest"/>
    <resultType xsi:type="pam-ops:StartResponse"/>
  </customOp>
  …
```

The schema for the connector operations is accessed through the REST API metadata interface at:

`http://<hostname>:7000/node/rest/CA:00074_CA:00074:01;metadata=schema`

Any third-party XML schema utility can use the schema to construct the input parameters of the operations.

For example:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://ns.ca.com/2011/09/pam-ops"
           xmlns:pam-ops="http://ns.ca.com/2011/09/pam-ops"
           xmlns:usm-meta="http://ns.ca.com/2009/07/usm-metadata"

xmlns:usm-meta2="http://ns.ca.com/2011/02/usm-metadata2"
           xmlns:usm-core="http://ns.ca.com/2009/07/usm-core" >

      <xs:import namespace="http://ns.ca.com/2009/07/usm-metadata"
schemaLocation="usm-metadata-200907.xsd"/>
      <xs:import
namespace="http://ns.ca.com/2011/02/usm-metadata2"
schemaLocation="usm-metadata2-201102.xsd"/>
      <xs:import namespace="http://ns.ca.com/2009/07/usm-core"
schemaLocation="usm-core-200907.xsd"/>


  <xs:complexType name="StartRequest">
    <xs:annotation>
      <xs:documentation>Request to start an entity, identified by its
EntityID</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="EntityID" type="usm-core:EntityID">
        <xs:annotation>
          <xs:documentation>Identification of the entity which
should be started (defined by the elements,
MdrProduct-MdrProdInstance-MdrElementID).</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="Arguments" type="pam-ops:ArgumentList"
minOccurs="0" maxOccurs="1" />
    </xs:sequence>
  </xs:complexType>
  <xs:element name="StartRequest" type="pam-ops:StartRequest">
    <xs:annotation>
      <xs:documentation>Request to start an entity, identified by its
EntityID</xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:complexType name="StartResponse">
    <xs:annotation>
      <xs:documentation>If successfully started, a response
containing the EntityID is returned</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="EntityID" type="usm-core:EntityID">
        <xs:annotation>
```

```
          <xs:documentation>Identification of the entity which was
requested to be started (defined by the elements,
MdrProduct-MdrProdInstance-MdrElementID).</xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="StartResponse" type="pam-ops:StartResponse">
    <xs:annotation>
      <xs:documentation>If successfully started, a response
containing theEntityID is returned</xs:documentation>
    </xs:annotation>
  </xs:element>
…
```

Starting with the URL of the Catalyst REST interface, you can display the Catalyst Process Automation Services URL. The Catalyst Process Automation Services URL contains the URLs of the supported types and operations. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="http://ns.ca.com/2010/11/coreapi">
  <id>CA:00074_CA:00074:01</id>
  <mdrProduct>CA:00074</mdrProduct>
  <mdrProdInstance> CA:00074:01</mdrProdInstance>
  <type>
    <id>ITActivity</id>
    <name>ITActivity</name>
    <namespace>http://ns.ca.com/2009/07/usm-core</namespace>
    <prefix/>
    <link
href="http://mulwi01-w500:7000/node/rest/CA:00074_CA:00074:01/ITAc
tivity" rel="list"/>
    <link
href="http://mulwi01-w500:7000/node/rest/CA:00074_CA:00074:01/ITAc
tivity;metadata" rel="metadata"/>
  </type>
  <type>
    <id>ITActivityTemplate</id>
    <name>ITActivityTemplate</name>
    <namespace>http://ns.ca.com/2009/07/usm-core</namespace>
    <prefix/>
    <link
href="http://mulwi01-w500:7000/node/rest/CA:00074_CA:00074:01/ITAc
tivityTemplate" rel="list"/>
    <link
href="http://mulwi01-w500:7000/node/rest/CA:00074_CA:00074:01/ITAc
tivityTemplate;metadata" rel="metadata"/>
  </type>
  <operation>
    <id>Release</id>
    <name>Release</name>
    <synchronous>true</synchronous>
    <link
href="http://mulwi01-w500:7000/node/rest/CA:00074_CA:00074:01/_ops
/Release" rel="self"/>
  </operation>
```

Display Processes

To display CA Process Automation processes in a browser, use the following URL:

```
http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/ITActivityTe
mplate
```

For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<enumeration xmlns="http://ns.ca.com/2010/11/coreapi">
  <link
href="http://mulwi01-w500:7000/node/rest/CA:00074_CA:00074:01/ITAc
tivityTemplate?count=-1&amp;start=0" rel="all"/>
  <usm-core:ITActivityTemplate
xmlns:usm-core="http://ns.ca.com/2009/07/usm-core">
    <usm-core:MdrProduct>CA:00074</usm-core:MdrProduct>
    <usm-core:MdrProdInstance>
CA:00074:01</usm-core:MdrProdInstance>
    <usm-core:MdrElementID>/Test/TestExec1</usm-core:MdrElementID>
    <usm-core:Label>Workflow:/Test/TestExec1:1</usm-core:Label>

<usm-core:CreationTimestamp>2011-11-28T13:07:03</usm-core:Creation
Timestamp>

<usm-core:LastModTimestamp>2011-11-28T13:07:19</usm-core:LastModTi
mestamp>
    <usm-core:InstanceName>Workflow:1</usm-core:InstanceName>

<usm-core:DefinitionName>/Test/TestExec1</usm-core:DefinitionName>
    <usm-core:DefinitionVersion>1</usm-core:DefinitionVersion>
    <usm-core:ActivityTypes>Workflow</usm-core:ActivityTypes>
    <link
href="http://mulwi01-w500:7000/node/rest/CA:00074_CA:00074:01/ITAc
tivityTemplate/Test/TestExec1" rel="self"/>
  </usm-core:ITActivityTemplate>
  <usm-core:ITActivityTemplate
xmlns:usm-core="http://ns.ca.com/2009/07/usm-core">
    <usm-core:MdrProduct>CA:00074</usm-core:MdrProduct>
    <usm-core:MdrProdInstance>
CA:00074:01</usm-core:MdrProdInstance>

<usm-core:MdrElementID>/Test/TestGetITActivity</usm-core:MdrElemen
tID>

<usm-core:Label>Workflow:/Test/TestGetITActivity:1</usm-core:Label
>

<usm-core:CreationTimestamp>2011-11-29T20:47:50</usm-core:Creation
Timestamp>

<usm-core:LastModTimestamp>2011-11-29T20:48:02</usm-core:LastModTi
mestamp>
    <usm-core:InstanceName>Workflow:1</usm-core:InstanceName>

<usm-core:DefinitionName>/Test/TestGetITActivity</usm-core:Definit
ionName>
```

```
    <usm-core:DefinitionVersion>1</usm-core:DefinitionVersion>
    <usm-core:ActivityTypes>Workflow</usm-core:ActivityTypes>
    <link
href="http://mulwi01-w500:7000/node/rest/CA:00074_CA:00074:01/ITAc
tivityTemplate/Test/TestGetITActivity" rel="self"/>
  </usm-core:ITActivityTemplate>
  <usm-core:ITActivityTemplate
xmlns:usm-core="http://ns.ca.com/2009/07/usm-core">
    <usm-core:MdrProduct>CA:00074</usm-core:MdrProduct>

<usm-core:MdrProdInstance>CA:00074:01</usm-core:MdrProdInstance>Md
rProdInstance>CA:00074:01</usm-core:MdrProdInstance>

<usm-core:MdrElementID>/Test/TestExecRelease</usm-core:MdrElementI
D>

<usm-core:Label>Workflow:/Test/TestExecRelease:1</usm-core:Label>

<usm-core:CreationTimestamp>2011-12-01T15:10:47</usm-core:Creation
Timestamp>

<usm-core:LastModTimestamp>2011-12-01T15:11:03</usm-core:LastModTi
mestamp>
    <usm-core:InstanceName>Workflow:1</usm-core:InstanceName>

<usm-core:DefinitionName>/Test/TestExecRelease</usm-core:Definitio
nName>
    <usm-core:DefinitionVersion>1</usm-core:DefinitionVersion>
    <usm-core:ActivityTypes>Workflow</usm-core:ActivityTypes>
    <link
href="http://mulwi01-w500:7000/node/rest/CA:00074_CA:00074:01/ITAc
tivityTemplate/Test/TestExecRelease" rel="self"/>
  </usm-core:ITActivityTemplate>
</enumeration>
```

Display Process Instances

To display CA Process Automation process instances in a browser, use the following URL:

```
http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/ITActivity
```

For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<enumeration xmlns="http://ns.ca.com/2010/11/coreapi">
  <link
href="http://mulwi01-w500:7000/node/rest/CA:00074_CA:00074:01/ITAc
tivity?count=-1&amp;start=0" rel="all"/>
  <usm-core:ITActivity
xmlns:usm-core="http://ns.ca.com/2009/07/usm-core">
    <usm-core:MdrProduct>CA:00074</usm-core:MdrProduct>

<usm-core:MdrProdInstance>CA:00074:01</usm-core:MdrProdInstance>Md
rProdInstance>CA:00074:01</usm-core:MdrProdInstance>
    <usm-core:MdrElementID>TestGetPAM_140</usm-core:MdrElementID>

<usm-core:Label>Workflow:TestGetPAM_140:140:140</usm-core:Label>

<usm-core:CreationTimestamp>2011-12-01T12:43:12</usm-core:Creation
Timestamp>

<usm-core:LastModTimestamp>2011-12-01T12:44:01</usm-core:LastModTi
mestamp>
    <usm-core:LastModUserName>pamadmin</usm-core:LastModUserName>

<usm-core:InstanceName>TestGetPAM_140:140:Workflow</usm-core:Insta
nceName>
    <usm-core:ActivityID>140</usm-core:ActivityID>
    <usm-core:RuntimeName>TestGetPAM_140</usm-core:RuntimeName>

<usm-core:RuntimeDiscriminator>140</usm-core:RuntimeDiscriminator>
    <usm-core:ActivityTypes>Workflow</usm-core:ActivityTypes>

<usm-core:ActivityState>Finished-Completed</usm-core:ActivityState
>

<usm-core:StateDescription>Completed</usm-core:StateDescription>
    <link
href="http://mulwi01-w500:7000/node/rest/CA:00074_CA:00074:01/ITAc
tivity/TestGetPAM_140" rel="self"/>
  </usm-core:ITActivity>
  <usm-core:ITActivity
xmlns:usm-core="http://ns.ca.com/2009/07/usm-core">
    <usm-core:MdrProduct>CA:00074</usm-core:MdrProduct>

<usm-core:MdrProdInstance>CA:00074:01</usm-core:MdrProdInstance>Md
rProdInstance>CA:00074:01</usm-core:MdrProdInstance>
    <usm-core:MdrElementID>TestGetPAM_135</usm-core:MdrElementID>

<usm-core:Label>Workflow:TestGetPAM_135:135:135</usm-core:Label>
```

```
<usm-core:CreationTimestamp>2011-12-01T11:59:53</usm-core:Creation
Timestamp>

<usm-core:LastModTimestamp>2011-12-01T12:01:11</usm-core:LastModTi
mestamp>
    <usm-core:LastModUserName>pamadmin</usm-core:LastModUserName>

<usm-core:InstanceName>TestGetPAM_135:135:Workflow</usm-core:Insta
nceName>
    <usm-core:ActivityID>135</usm-core:ActivityID>
    <usm-core:RuntimeName>TestGetPAM_135</usm-core:RuntimeName>

<usm-core:RuntimeDiscriminator>135</usm-core:RuntimeDiscriminator>
    <usm-core:ActivityTypes>Workflow</usm-core:ActivityTypes>

<usm-core:ActivityState>Finished-Failed</usm-core:ActivityState>
    <usm-core:StateDescription>Failed</usm-core:StateDescription>
    <link
href="http://mulwi01-w500:7000/node/rest/CA:00074_CA:00074:01/ITAc
tivity/TestGetPAM_135" rel="self"/>
  </usm-core:ITActivity>
  <usm-core:ITActivity
xmlns:usm-core="http://ns.ca.com/2009/07/usm-core">
    <usm-core:MdrProduct>CA:00074</usm-core:MdrProduct>

<usm-core:MdrProdInstance>CA:00074:01</usm-core:MdrProdInstance>Md
rProdInstance>CA:00074:01</usm-core:MdrProdInstance>

<usm-core:MdrElementID>TestGetITActivity_130</usm-core:MdrElementI
D>

<usm-core:Label>Workflow:TestGetITActivity_130:130:130</usm-core:L
abel>

<usm-core:CreationTimestamp>2011-12-01T11:53:44</usm-core:Creation
Timestamp>

<usm-core:LastModTimestamp>2011-12-01T11:59:14</usm-core:LastModTi
mestamp>
    <usm-core:LastModUserName>pamadmin</usm-core:LastModUserName>

<usm-core:InstanceName>TestGetITActivity_130:130:Workflow</usm-cor
e:InstanceName>
    <usm-core:ActivityID>130</usm-core:ActivityID>

<usm-core:RuntimeName>TestGetITActivity_130</usm-core:RuntimeName>

<usm-core:RuntimeDiscriminator>130</usm-core:RuntimeDiscriminator>
```

```
    <usm-core:ActivityTypes>Workflow</usm-core:ActivityTypes>

<usm-core:ActivityState>Finished-Completed</usm-core:ActivityState
>

<usm-core:StateDescription>Completed</usm-core:StateDescription>
    <link
href="http://mulwi01-w500:7000/node/rest/CA:00074_CA:00074:01/ITAc
tivity/TestGetITActivity_130" rel="self"/>
  </usm-core:ITActivity>
</enumeration>
```

Display Process Relationships

To display CA Process Automation process relationships in a browser, use the following URL:

```
http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/BinaryRelati
onship
```

For example:

```
<?xml version="1.0" encoding="utf-8"?>
<enumeration xmlns="http://ns.ca.com/2010/11/coreapi">
  <link
href="http://mulwi01-w500:7000/node/rest/CA:00074_CA:0074_CA:0074:
01/BinaryRelationship?count=-1&amp;start=0" rel="all"/>
  <usm-core:BinaryRelationship
xmlns:usm-core="http://ns.ca.com/2009/07/usm-core">
    <usm-core:MdrProduct>CA:00074</usm-core:MdrProduct>

<usm-core:MdrProdInstance>CA:00074:01</usm-core:MdrProdInstance>

<usm-core:MdrElementID>TestGetITActivityTemplate_1</usm-core:MdrEl
ementID>
    <usm-core:Label>IsInstanceOf</usm-core:Label>
    <usm-core:InstanceName/>

<usm-core:SourceMdrProduct>CA:00074</usm-core:SourceMdrProduct>

<usm-core:SourceMdrProdInstance>CA:0074_CA:0074:01</usm-core:Sourc
eMdrProdInstance>

<usm-core:SourceMdrElementID>TestGetITActivityTemplate_1</usm-core
:SourceMdrElementID>

<usm-core:TargetMdrProduct>CA:00074</usm-core:TargetMdrProduct>

<usm-core:TargetMdrProdInstance>CA:0074_CA:0074:01</usm-core:Targe
tMdrProdInstance>

<usm-core:TargetMdrElementID>/Tests/TestGetITActivityTemplate</usm
-core:TargetMdrElementID>
    <usm-core:Semantic>IsInstanceOf</usm-core:Semantic>
    <link
href="http://mulwi01-w500:7000/node/rest/CA:00074_CA:0074_CA:0074:
01/BinaryRelationship/TestGetITActivityTemplate_1" rel="self"/>
  </usm-core:BinaryRelationship>
  <usm-core:BinaryRelationship
xmlns:usm-core="http://ns.ca.com/2009/07/usm-core">
    <usm-core:MdrProduct>CA:00074</usm-core:MdrProduct>

<usm-core:MdrProdInstance>CA:00074:01</usm-core:MdrProdInstance>

<usm-core:MdrElementID>TestGetITActivity_1</usm-core:MdrElementID>
    <usm-core:Label>IsInstanceOf</usm-core:Label>
    <usm-core:InstanceName/>

<usm-core:SourceMdrProduct>CA:00074</usm-core:SourceMdrProduct>
```

```
<usm-core:SourceMdrProdInstance>CA:0074_CA:0074:01</usm-core:Sourc
eMdrProdInstance>

<usm-core:SourceMdrElementID>TestGetITActivity_1</usm-core:SourceM
drElementID>

<usm-core:TargetMdrProduct>CA:00074</usm-core:TargetMdrProduct>

<usm-core:TargetMdrProdInstance>CA:0074_CA:0074:01</usm-core:Targe
tMdrProdInstance>

<usm-core:TargetMdrElementID>/Tests/TestGetITActivity</usm-core:Ta
rgetMdrElementID>
    <usm-core:Semantic>IsInstanceOf</usm-core:Semantic>
    <link
href="http://mulwi01-w500:7000/node/rest/CA:00074_CA:0074_CA:0074:
01/BinaryRelationship/TestGetITActivity_1" rel="self"/>
  </usm-core:BinaryRelationship>
</enumeration>
```

Display a Specific Object

You can display any specific object in a browser using these URL formats:

`http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/<type>/<id>`

`http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/<type>?id=<id>`

Where:

- <*type*> is the USM type
- <*id*> is the MdrElementID value

For example, use the following URL to display a process instance (ITActivity) with id of "TestSubscribeAlert_59":

`http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/ITActivity/TestSubscribeAlert_59`

`http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/ITActivity?id=TestSubscribeAlert_59`

By refreshing the browser, you can monitor the process instance status (in the ActivityState or StateDescription properties of the ITActivity).

```
<?xml version="1.0" encoding="utf-8"?>
<usm-core:ITActivity
xmlns:usm-core="http://ns.ca.com/2009/07/usm-core">
  <usm-core:MdrProduct>CA:00074</usm-core:MdrProduct>
  <usm-core:MdrProdInstance>CA:00074:01</usm-core:MdrProdInstance>

<usm-core:MdrElementID>TestSubscribeAlert_59</usm-core:MdrElementI
D>

<usm-core:Label>Workflow:TestSubscribeAlert_59:59:59</usm-core:Lab
el>

<usm-core:CreationTimestamp>2012-04-18T15:57:38</usm-core:Creation
Timestamp>

<usm-core:LastModTimestamp>2012-04-18T15:59:18</usm-core:LastModTi
mestamp>
  <usm-core:LastModUserName>pamadmin</usm-core:LastModUserName>

<usm-core:InstanceName>TestSubscribeAlert_59:59:Workflow</usm-core
:InstanceName>
  <usm-core:ActivityID>59</usm-core:ActivityID>

<usm-core:RuntimeName>TestSubscribeAlert_59</usm-core:RuntimeName>

<usm-core:RuntimeDiscriminator>59</usm-core:RuntimeDiscriminator>
  <usm-core:ActivityTypes>Workflow</usm-core:ActivityTypes>

<usm-core:ActivityState>Finished-Completed</usm-core:ActivityState
>
  <usm-core:StateDescription>Completed</usm-core:StateDescription>
  <link xmlns="http://ns.ca.com/2010/11/coreapi"
href="http://mulwi01-w500:7000/node/rest/CA:00074_CA:00074:01/ITAc
tivity/TestSubscribeAlert_59" rel="self"/>
</usm-core:ITActivity>
```

Execute Connector Operations

To run connector operations, send HTTP POST requests using the operation URL.

Note: Only the execute operations require POST requests. All other requests use GET.

For example, to start a process, send an HTTP POST request to the following URL:

`http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/_ops/Start`

Include a request header:

`Content-Type=application/xml`

Include a request body that contains a StartRequest operation that includes the following items:

- The process name in the MdrElementID element
- The parameters in the Arguments element

For example:

```
<StartRequest
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="p1:StartRequest"
xmlns:p1="http://ns.ca.com/2011/09/pam-ops" >
      <EntityID>

<MdrElementID>/Tests/TestGetITActivityTemplate</MdrElementID>
      </EntityID>
      <Arguments>
        <Argument>
          <Name>arg1</Name>
          <Value>val1</Value>
        </Argument>
        <Argument>
          <Name>arg2</Name>
          <Value>val2</Value>
        </Argument>
        <Argument>
          <Name>arg3</Name>
          <Value>val3</Value>
        </Argument>
      </Arguments>
</StartRequest>
```

Start Process

You can use any HTTP Client tool or application to send the request to start the process.

For example, to start a process, send an HTTP POST request to the following URL:

`http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/_ops/Start`

Include a request header:

```
Content-Type=application/xml
```

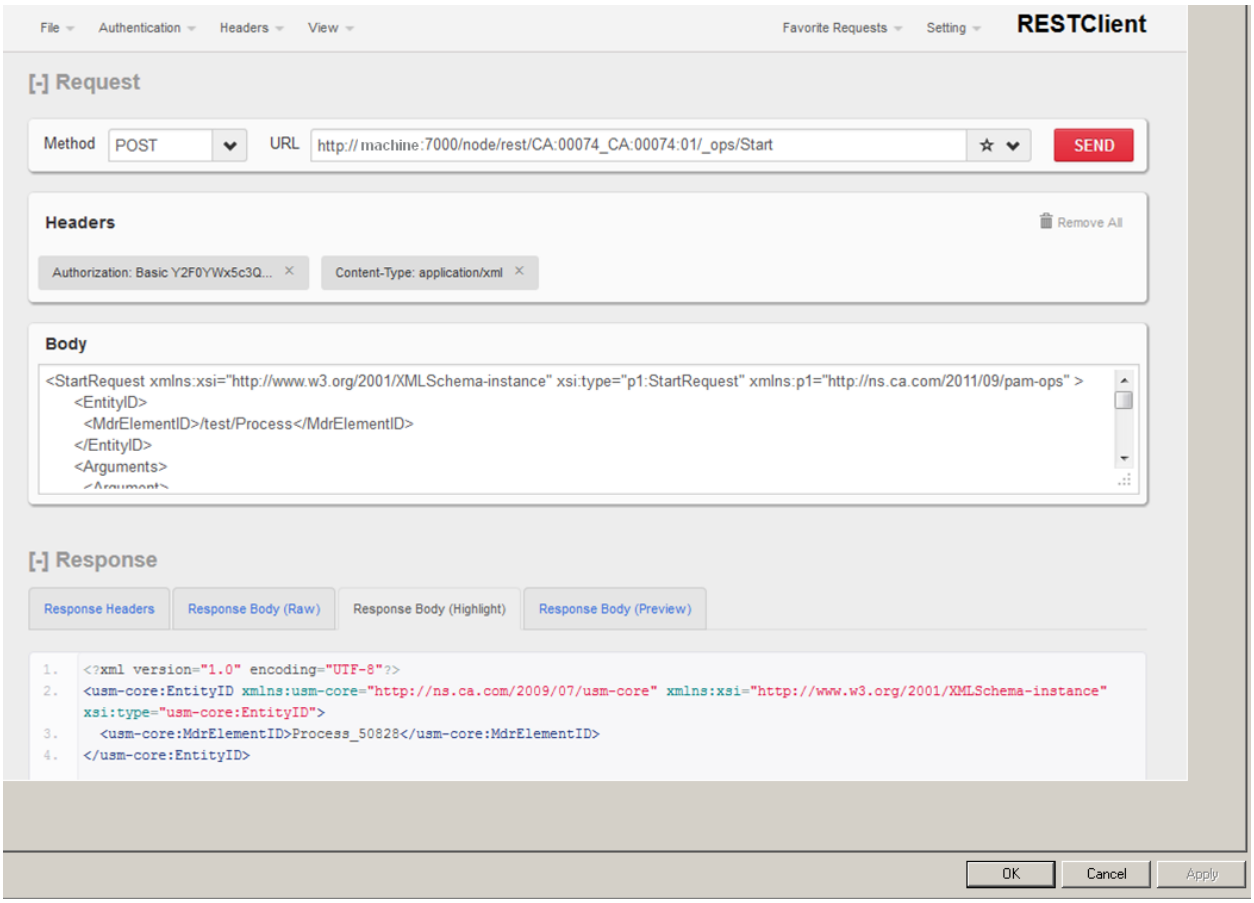
For example:

```
<StartRequest
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="p1:StartRequest"
xmlns:p1="http://ns.ca.com/2011/09/pam-ops" >
      <EntityID>
        <MdrElementID>/test/Process</MdrElementID>
      </EntityID>
      <Arguments>
        <Argument>
          <Name>arg1</Name>
          <Value>val1</Value>
        </Argument>
        <Argument>
          <Name>arg2</Name>
          <Value>val2</Value>
        </Argument>
        <Argument>
          <Name>arg3</Name>
          <Value>val3</Value>
        </Argument>
      </Arguments>
    </StartRequest>
```

The request returns the following XML:

```
<usm-core:EntityID xsi:type="usm-core:EntityID">
<usm-core:MdrElementID>Process_51268</usm-core:MdrElementID>
</usm-core:EntityID>
```

The Firefox REST Client is shown in this example:



Hold

To hold (or suspend) a process, specify the process ID in the MdrElementID property.

For example, to hold a process, send an HTTP POST request to the following URL:

`http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/_ops/Hold`

Include a request header:

```
Content-Type=application/xml
```

For example:

```
<HoldRequest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:type="p1:HoldRequest"
             xmlns:p1="http://ns.ca.com/2011/09/pam-ops" >
  <EntityID>
    <MdrProduct/>
    <MdrProdInstance/>
    <MdrElementID>TestIRFPrompt_140</MdrElementID>
  </EntityID>
</HoldRequest>
```

The request returns the following XML:

```
<usm-core:EntityID
xmlns:usm-core="http://ns.ca.com/2009/07/usm-core"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                   xsi:type="usm-core:EntityID">
  <usm-core:MdrProduct/>
  <usm-core:MdrProdInstance/>
  <usm-core:MdrElementID>TestIRFPrompt_140</usm-core:MdrElementID>
</usm-core:EntityID>
```

Release

To release (or resume) a process, specifying the process ID in the MdrElementID property.

For example, to release a process, send an HTTP POST request to the following URL:

```
http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/_ops/Release
```

Include a request header:

```
Content-Type=application/xml
```

For example:

```
<ReleaseRequest
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:type="p1:ReleaseRequest"
                xmlns:p1="http://ns.ca.com/2011/09/pam-ops" >
  <EntityID>
    <MdrProduct/>
    <MdrProdInstance/>
    <MdrElementID>TestIRFPrompt_140</MdrElementID>
  </EntityID>
</ReleaseRequest>
```

The request returns the following XML:

```
<usm-core:EntityID
xmlns:usm-core="http://ns.ca.com/2009/07/usm-core"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:type="usm-core:EntityID">
  <usm-core:MdrProduct/>
  <usm-core:MdrProdInstance/>
  <usm-core:MdrElementID>TestIRFPrompt_140</usm-core:MdrElementID>
</usm-core:EntityID>
```

Cancel

To cancel (or stop) a process, specify the process ID in the MdrElementID property.

For example, to cancel a process, send an HTTP POST request to the following URL:

```
http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/_ops/Cancel
```

Include a request header:

```
Content-Type=application/xml
```

For example:

```
<CancelRequest
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:type="p1:CancelRequest"
               xmlns:p1="http://ns.ca.com/2011/09/pam-ops" >
  <EntityID>
    <MdrProduct/>
    <MdrProdInstance/>
    <MdrElementID>TestIRFPrompt_140</MdrElementID>
  </EntityID>
</CancelRequest>
```

The request returns the following XML:

```
<usm-core:EntityID
xmlns:usm-core="http://ns.ca.com/2009/07/usm-core"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                   xsi:type="usm-core:EntityID">
  <usm-core:MdrProduct/>
  <usm-core:MdrProdInstance/>
  <usm-core:MdrElementID>TestIRFPrompt_140</usm-core:MdrElementID>
</usm-core:EntityID>
```

Start Request REST Examples

The following examples demonstrate how to use RESTful services with start request forms.

QueryStartRequestForms

You can use an optional filter with a library path to query start request forms.

For example, send an HTTP POST request to:

```
http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/_ops/QuerySt
artRequestForms
```

Include a request header:

```
Content-Type=application/xml
```

For example:

```
<QueryStartRequestFormsRequest
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:type="p1:QueryStartRequestFormsRequest"

xmlns:p1="http://ns.ca.com/2011/09/pam-ops" >
  <Filter>
    <LookUpPath>/TestSRF</LookUpPath>
    <IsRecursive>true</IsRecursive>
        <Keywords>
      <Keyword>
        <Name>TagValue</Name>
      </Keyword>
    </Keywords>
  </Filter>
</QueryStartRequestFormsRequest>
```

As the following example shows, the response from QueryStartRequestForms includes the start request form parameter names and types (shown in **bold** in the example). The SubmitStartRequestForm operation uses the parameter names and types.

```
<?xml version="1.0" encoding="utf-8"?>
<pam-ops:QueryStartRequestFormsResponse
xmlns:pam-ops="http://ns.ca.com/2011/09/pam-ops"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:type="pam-ops:QueryStartRequestFormsResponse">
  <StartRequestForms>
    <StartRequestForm>
      <Name>SRF</Name>
      <Path>/TestSRF/</Path>
      <Description>my test srf description</Description>
      <Pages>
        <HasComplexType>false</HasComplexType>
        <Page>
          <Name>TestPage</Name>
          <Param>
            <Name>Var_0</Name>
            <Label>Text Field</Label>
            <Description/>
            <IsReadOnly>false</IsReadOnly>
            <Type>String</Type>
          </Param>
          <Param>
            <Name>Var_1</Name>
            <Label>Check Box</Label>
            <Description/>
            <IsReadOnly>false</IsReadOnly>
            <Type>Boolean</Type>
          </Param>
          <Param>
            <Name>Var_2</Name>
            <Label>Select</Label>
            <Description/>
            <IsReadOnly>false</IsReadOnly>
            <Type>String</Type>
          </Param>
        </Page>
        <Page>
          <Name>System</Name>
        </Page>
      </Pages>
    </StartRequestForm>
  </StartRequestForms>
</pam-ops:QueryStartRequestFormsResponse>
```

SubmitStartRequestForm

To submit a start request form, use the parameters defined in the QueryStartRequestForms response.

For example, send an HTTP POST request to:

`http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/_ops/SubmitStartRequestForm`

Include a request header:

`Content-Type=application/xml`

For example:

```
<SubmitStartRequestFormRequest
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:type="p1:SubmitStartRequestFormRequest"

xmlns:p1="http://ns.ca.com/2011/09/pam-ops" >

  <Name>SRF</Name>

  <Path>/TestSRF</Path>

  <Arguments>

    <Argument>
      <Name>Var_0</Name>
      <Value>val0</Value>

    </Argument>

    <Argument>
      <Name>Var_1</Name>
      <Value>true</Value>

    </Argument>

    <Argument>
      <Name>Var_2</Name>
      <Value>val2</Value>

    </Argument>

  </Arguments>

  <Options>

    <StartDate>startDate</StartDate>

    <StartTime>startTime</StartTime>

  </Options>

</SubmitStartRequestFormRequest>
```

Note:

Use the following format for StartDate and StartTime:

- Start Date: yyyy-MM-dd
- StartTime: HH:mm (24-hour format)

As the following example shows, the response from SubmitStartRequestForm contains the start request identifier (shown in **bold** in the example):

```
<?xml version="1.0" encoding="utf-8"?>
<pam-ops:SubmitStartRequestFormResponse
xmlns:pam-ops="http://ns.ca.com/2011/09/pam-ops"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:type="pam-ops:SubmitStartRequestFormResponse">
  <StartRequestStatus>
    <InteractionID>1291</InteractionID>
    <Instance>SRF_1291</Instance>
    <State>Queued</State>

<Earliest-start-time>2012-06-14T15:52:05.341Z</Earliest-start-time
>
    <RefProcess>/TestSRF/TestProcess</RefProcess>
  </StartRequestStatus>
</pam-ops:SubmitStartRequestFormResponse>
```

QueryStartRequests

You can query start requests using an optional filter.

For example, send an HTTP POST request to:

```
http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/_ops/QuerySt
artRequests
```

Include a request header:

```
Content-Type=application/xml
```

For example:

```
<QueryStartRequestsRequest
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                            xsi:type="p1:QueryStartRequestsRequest"

xmlns:p1="http://ns.ca.com/2011/09/pam-ops" >
<Instance>SRF_1291</Instance>
<IsArchived>false</IsArchived>
</QueryStartRequestsRequest>
```

As the following example shows, the response from QueryStartRequests contains the start request instances:

```
<?xml version="1.0" encoding="utf-8"?>
<pam-ops:QueryStartRequestsResponse
xmlns:pam-ops="http://ns.ca.com/2011/09/pam-ops"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:type="pam-ops:QueryStartRequestsResponse">
  <StartRequestInstances>
    <StartRequestInstance>
      <Instance>SRF_1291</Instance>
      <ProcessInstance>TestProcess_1292</ProcessInstance>
      <State>Completed</State>
      <Scheduled-time>2012-06-14T15:52:05.000Z</Scheduled-time>
      <Start-time>2012-06-14T15:52:05.000Z</Start-time>
      <End-time>2012-06-14T15:52:11.000Z</End-time>
    </StartRequestInstance>
  </StartRequestInstances>
</pam-ops:QueryStartRequestsResponse>
```

DequeueStartRequest

You can dequeue a queued start request.

For example, send an HTTP POST request to:

```
http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/_ops/Dequeue
StartRequest
```

Include a request header:

```
Content-Type=application/xml
```

For example:

```
<DequeueStartRequestRequest
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:type="p1:DequeueStartRequestRequest"

xmlns:p1="http://ns.ca.com/2011/09/pam-ops">
  <Instance>SRF_1_44</Instance>
</DequeueStartRequestRequest>
```

The following example shows the response from DequeueStartRequest:

```
<?xml version="1.0" encoding="UTF-8"?>
<pam-ops:DequeueStartRequestResponse
xmlns:pam-ops="http://ns.ca.com/2011/09/pam-ops"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:type="pam-ops:DequeueStartRequestResponse">
  <StartRequestInstance>
    <Instance>SRF_1_44</Instance>
    <ProcessInstance xsi:nil="true"/>
    <State>Failed</State>
    <Scheduled-time>2012-07-01T13:39:00.000Z</Scheduled-time>
  </StartRequestInstance>
</pam-ops:DequeueStartRequestResponse>
```

AbortStartRequest

Use AbortStartRequest to end a running start request.

For example, send an HTTP POST request to:

http://<*hostname*>:7000/node/rest/CA:00074_CA:00074:01/_ops/AbortStartRequest

Include a request header:

```
Content-Type=application/xml
```

For example:

```
<AbortStartRequestRequest
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                           xsi:type="p1:AbortStartRequestRequest"

xmlns:p1="http://ns.ca.com/2011/09/pam-ops">
  <Instance>SRF_1_38</Instance>
</AbortStartRequestRequest>
```

The following example shows the AbortStartRequest response:

```
<?xml version="1.0" encoding="utf-8"?>
<pam-ops:AbortStartRequestResponse
xmlns:pam-ops="http://ns.ca.com/2011/09/pam-ops"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:type="pam-ops:AbortStartRequestResponse">
  <StartRequestInstance>
    <Instance>SRF_1_38</Instance>
    <ProcessInstance>TestProcess_39</ProcessInstance>
    <State>Failed</State>
    <Scheduled-time>2012-06-22T13:28:42.000Z</Scheduled-time>
    <Start-time>2012-06-22T13:28:42.000Z</Start-time>
  </StartRequestInstance>
</pam-ops:AbortStartRequestResponse>
```

Task REST Operations Examples

The following examples demonstrate how to use RESTful services to conduct various task actions.

QueryTasks

You can query a specific task or all tasks, pending interaction requests.

- To query a specific task, provide a TaskID tag.
- To query all tasks, do not specify a TaskID tag.

For example, send an HTTP POST request to:

```
http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/_ops/QueryTa
sks
```

Include a request header:

```
Content-Type=application/xml
```

For example:

```
<QueryTasksRequest
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:type="p1:QueryTasksRequest"
                  xmlns:p1="http://ns.ca.com/2011/09/pam-ops" >
    <TaskID>510</TaskID>
</QueryTasksRequest>
```

The response contains the Task ID, parameter names, Assignees, and Delegates. Use the parameter names and types in the ReplyTask operation (shown in **bold** in the example).

```
<?xml version="1.0" encoding="utf-8"?>
<pam-ops:QueryTasksResponse
xmlns:pam-ops="http://ns.ca.com/2011/09/pam-ops"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                              xsi:type="pam-ops:QueryTasksResponse">
  The topics that follow describe how to perform the tasks most routine
to this interface.
    <Task>
      <TaskID>510</TaskID>
      <ProcessID>505</ProcessID>
      <State>Completed</State>
      <Instance>TestIRFPrompt_505</Instance>
      <Title>Assign title</Title>
      <Description>Assign description 1</Description>
      <StartTime>2012-06-01T17:37:08.000Z</StartTime>
      <RootUUID>eba34640-5de1-4e73-ab28-e312a1ef35fa</RootUUID>
      <IsApprovalRequired>true</IsApprovalRequired>
      <Pages>
        <HasComplexType>false</HasComplexType>
        <Page>
          <Name>TestPage</Name>
          <Param>
            <Name>Var_0</Name>
            <Label>TextField1</Label>
            <Description/>
            <IsReadOnly>false</IsReadOnly>
            <Type>String</Type>
          </Param>
          <Param>
            <Name>Var_1</Name>
            <Label>Check Box</Label>
            <Description/>
            <IsReadOnly>false</IsReadOnly>
            <Type>Boolean</Type>
          </Param>
          <Param>
            <Name>Var_2</Name>
            <Label>Select</Label>
            <Description/>
            <IsReadOnly>false</IsReadOnly>
            <Type>String</Type>
          </Param>
        </Page>
        <Page>
          <Name>User Prompt</Name>
        </Page>
```

```
          <Page>
            <Name>System</Name>
          </Page>
        </Pages>
        <Assignees>
          <Users>
            <User>
              <Name>pamadmin</Name>
            </User>
          </Users>
        </Assignees>
      </Task>
    </Tasks>
</pam-ops:QueryTasksResponse>
```

ReplyTask

You can reply to a task with parameter values and approval (shown in bold in the following example).

For example, send an HTTP POST request to:

`http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/_ops/ReplyTask`

Include a request header:

```
Content-Type=application/xml
```

For example:

```
<ReplyTaskRequest
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                  xsi:type="p1:ReplyTaskRequest"
                  xmlns:p1="http://ns.ca.com/2011/09/pam-ops" >
  <TaskID>510</TaskID>
  <IsApproved>true</IsApproved>
  <Arguments>
    <Argument>
      <Name>Var_0</Name>
      <Value>val0</Value>
    </Argument>
    <Argument>
      <Name>Var_1</Name>
      <Value>true</Value>
    </Argument>
    <Argument>
      <Name>Var_2</Name>
      <Value>val2</Value>
    </Argument>
  </Arguments>
</ReplyTaskRequest>
```

The following example shows the response from ReplyTask:

```
<?xml version="1.0" encoding="utf-8"?>
<pam-ops:ReplyTaskResponse
xmlns:pam-ops="http://ns.ca.com/2011/09/pam-ops"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                            xsi:type="pam-ops:ReplyTaskResponse">
  <Task>
    <TaskID>510</TaskID>
    <ProcessID>505</ProcessID>
    <State>Completed</State>
    <Instance>TestIRFPrompt_505</Instance>
    <Title>Assign title</Title>
    <Description>Assign description 1</Description>
    <StartTime>2012-06-01T17:37:08.000Z</StartTime>
    <RootUUID>eba34640-5de1-4e73-ab28-e312a1ef35fa</RootUUID>
    <IsApprovalRequired>true</IsApprovalRequired>
    <Pages>
      <HasComplexType>false</HasComplexType>
      <Page>
        <Name>TestPage</Name>
        <Param>
          <Name>Var_0</Name>
          <Label>TextField1</Label>
          <Description/>
          <IsReadOnly>false</IsReadOnly>
          <Type>String</Type>
        </Param>
        <Param>
          <Name>Var_1</Name>
          <Label>Check Box</Label>
          <Description/>
          <IsReadOnly>false</IsReadOnly>
          <Type>Boolean</Type>
        </Param>
        <Param>
          <Name>Var_2</Name>
          <Label>Select</Label>
          <Description/>
          <IsReadOnly>false</IsReadOnly>
          <Type>String</Type>
        </Param>
      </Page>
      <Page>
        <Name>User Prompt</Name>
      </Page>
      <Page>
        <Name>System</Name>
      </Page>
```

```
    </Pages>
    <Assignees>
      <Users>
        <User>
          <Name>pamadmin</Name>
        </User>
      </Users>
    </Assignees>
    <Delegates>
      <Users>
        <User>
          <Name>pamadmin</Name>
        </User>
      </Users>
    </Delegates>
  </Task>
</pam-ops:ReplyTaskResponse>
```

TakeTask

You can take a task by Task ID.

For example, send an HTTP POST request to:

`http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/_ops/TakeTask`

Include a request header:

`Content-Type=application/xml`

For example:

```
<TakeTaskRequest
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

                  xsi:type="p1:TakeTaskRequest"

                  xmlns:p1="http://ns.ca.com/2011/09/pam-ops" >

  <TaskID>510</TaskID>

</TakeTaskRequest>
```

The response:

```
<?xml version="1.0" encoding="utf-8"?>
<pam-ops:TakeTaskResponse
xmlns:pam-ops="http://ns.ca.com/2011/09/pam-ops"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                          xsi:type="pam-ops:TakeTaskResponse">
  <Task>
    <TaskID>510</TaskID>
    <ProcessID>505</ProcessID>
```

<State>eiamCertKeyPath

Specifies the name of the security certificate key file that is used for authentication. This property is only applicable if isFipsMode=true.

Example

```
<?xml version="1.0" encoding="utf-8"?>
<pam-ops:TakeTaskResponse
xmlns:pam-ops="http://ns.ca.com/2011/09/pam-ops"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                            xsi:type="pam-ops:TakeTaskResponse">
  <Task>
    <TaskID>510</TaskID>
    <ProcessID>505</ProcessID>
    <State>Taken</State>
    <Instance>TestIRFPrompt_505</Instance>
    <Title>Assign title</Title>
    <Description>Assign description 1</Description>
    <StartTime>2012-06-01T17:37:08.000Z</StartTime>
    <RootUUID>eba34640-5de1-4e73-ab28-e312a1ef35fa</RootUUID>
    <IsApprovalRequired>true</IsApprovalRequired>
    <Pages>
      <HasComplexType>false</HasComplexType>
      <Page>
        <Name>TestPage</Name>
        <Param>
          <Name>Var_0</Name>
          <Label>TextField1</Label>
          <Description/>
          <IsReadOnly>false</IsReadOnly>
          <Type>String</Type>
        </Param>
        <Param>
          <Name>Var_1</Name>
          <Label>Check Box</Label>
          <Description/>
          <IsReadOnly>false</IsReadOnly>
          <Type>Boolean</Type>
```

```
            </Param>
            <Param>
              <Name>Var_2</Name>
              <Label>Select</Label>
              <Description/>
              <IsReadOnly>false</IsReadOnly>
              <Type>String</Type>
            </Param>
          </Page>
          <Page>
            <Name>User Prompt</Name>
          </Page>
          <Page>
            <Name>System</Name>
          </Page>
        </Pages>
        <Assignees>
          <Users>
            <User>
              <Name>pamadmin</Name>
            </User>
          </Users>
        </Assignees>
      </Task>
    </pam-ops:TakeTaskResponse>
```

ReturnTask

You can return a task by Task ID.

For example, send an HTTP POST request to:

`http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/_ops/ReturnTask`

Include a request header:

`Content-Type=application/xml`

For example:

```
<ReturnTaskRequest
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

                    xsi:type="p1:ReturnTaskRequest"

                    xmlns:p1="http://ns.ca.com/2011/09/pam-ops" >

  <TaskID>510</TaskID>

</ReturnTaskRequest>
```

The response:

```
<?xml version="1.0" encoding="utf-8"?>
<pam-ops:ReturnTaskResponse
xmlns:pam-ops="http://ns.ca.com/2011/09/pam-ops"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                              xsi:type="pam-ops:ReturnTaskResponse">
  <Task>
    <TaskID>510</TaskID>
    <ProcessID>505</ProcessID>
    <State>Pending</State>
    <Instance>TestIRFPrompt_505</Instance>
    <Title>Assign title</Title>
    <Description>Assign description 1</Description>
    <StartTime>2012-06-01T17:37:08.000Z</StartTime>
    <RootUUID>eba34640-5de1-4e73-ab28-e312a1ef35fa</RootUUID>
    <IsApprovalRequired>true</IsApprovalRequired>
    <Pages>
      <HasComplexType>false</HasComplexType>
      <Page>
        <Name>TestPage</Name>
        <Param>
          <Name>Var_0</Name>
          <Label>TextField1</Label>
          <Description/>
          <IsReadOnly>false</IsReadOnly>
          <Type>String</Type>
        </Param>
        <Param>
          <Name>Var_1</Name>
          <Label>Check Box</Label>
          <Description/>
          <IsReadOnly>false</IsReadOnly>
          <Type>Boolean</Type>
        </Param>
        <Param>
          <Name>Var_2</Name>
          <Label>Select</Label>
          <Description/>
          <IsReadOnly>false</IsReadOnly>
          <Type>String</Type>
        </Param>
      </Page>
      <Page>
        <Name>User Prompt</Name>
      </Page>
      <Page>
        <Name>System</Name>
      </Page>
```

```
    </Pages>
    <Assignees>
```

DelegateTask

For example, send an HTTP POST request to:

`http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/_ops/DelegateTask`

Include a request header:

`Content-Type=application/xml`

For example:

```
<DelegateTaskRequest
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                     xsi:type="p1:DelegateTaskRequest"
                     xmlns:p1="http://ns.ca.com/2011/09/pam-ops" >
  <TaskID>510</TaskID>
  <Users>
    <User>
      <Name>pamadmin</Name>
    </User>
  </Users>
  <Groups>
    <Group/>
  </Groups>
</DelegateTaskRequest>
```

The response:

```
<?xml version="1.0" encoding="utf-8"?>
<pam-ops:DelegateTaskResponse
xmlns:pam-ops="http://ns.ca.com/2011/09/pam-ops"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:type="pam-ops:DelegateTaskResponse">
  <Task>
    <TaskID>510</TaskID>
    <ProcessID>505</ProcessID>
    <State>Pending</State>
    <Instance>TestIRFPrompt_505</Instance>
    <Title>Assign title</Title>
    <Description>Assign description 1</Description>
    <StartTime>2012-06-01T17:37:08.000Z</StartTime>
    <RootUUID>eba34640-5de1-4e73-ab28-e312a1ef35fa</RootUUID>
    <IsApprovalRequired>true</IsApprovalRequired>
    <Pages>
      <HasComplexType>false</HasComplexType>
      <Page>
        <Name>TestPage</Name>
        <Param>
          <Name>Var_0</Name>
          <Label>TextField1</Label>
          <Description/>
          <IsReadOnly>false</IsReadOnly>
          <Type>String</Type>
        </Param>
        <Param>
          <Name>Var_1</Name>
          <Label>Check Box</Label>
          <Description/>
          <IsReadOnly>false</IsReadOnly>
          <Type>Boolean</Type>
        </Param>
        <Param>
          <Name>Var_2</Name>
          <Label>Select</Label>
          <Description/>
          <IsReadOnly>false</IsReadOnly>
          <Type>String</Type>
        </Param>
      </Page>
      <Page>
        <Name>User Prompt</Name>
      </Page>
      <Page>
```

TransferTask

You can transfer (assign) a task by Task ID to one or more users and one or more groups.

For example, send an HTTP POST request to:

```
http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/_ops/TransferTask
```

Include a request header:

```
Content-Type=application/xml
```

For example:

```
<TransferTaskRequest
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                     xsi:type="p1:TransferTaskRequest"
                     xmlns:p1="http://ns.ca.com/2011/09/pam-ops" >
  <TaskID>510</TaskID>
  <Users>
    <User>
      <Name>pamadmin</Name>
    </User>
  </Users>
  <Groups>
    <Group/>
  </Groups>
</TransferTaskRequest>
```

The TransferTask request returns the following XML:

```
<?xml version="1.0" encoding="utf-8"?>
<pam-ops:TransferTaskResponse
xmlns:pam-ops="http://ns.ca.com/2011/09/pam-ops"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:type="pam-ops:TransferTaskResponse">
  <Task>
    <TaskID>510</TaskID>
    <ProcessID>505</ProcessID>
    <State>Pending</State>
    <Instance>TestIRFPrompt_505</Instance>
    <Title>Assign title</Title>
    <Description>Assign description 1</Description>
    <StartTime>2012-06-01T17:37:08.000Z</StartTime>
    <RootUUID>eba34640-5de1-4e73-ab28-e312a1ef35fa</RootUUID>
    <IsApprovalRequired>true</IsApprovalRequired>
    <Pages>
      <HasComplexType>false</HasComplexType>
      <Page>
        <Name>TestPage</Name>
        <Param>
          <Name>Var_0</Name>
          <Label>TextField1</Label>
          <Description/>
          <IsReadOnly>false</IsReadOnly>
          <Type>String</Type>
        </Param>
        <Param>
          <Name>Var_1</Name>
          <Label>Check Box</Label>
          <Description/>
          <IsReadOnly>false</IsReadOnly>
          <Type>Boolean</Type>
        </Param>
        <Param>
          <Name>Var_2</Name>
          <Label>Select</Label>
          <Description/>
          <IsReadOnly>false</IsReadOnly>
          <Type>String</Type>
        </Param>
      </Page>
      <Page>
        <Name>User Prompt</Name>
      </Page>
      <Page>
        <Name>System</Name>
```

```
      </Page>
    </Pages>
    <Assignees>
      <Users>
        <User>
          <Name>pamadmin</Name>
        </User>
      </Users>
    </Assignees>
  </Task>
</pam-ops:TransferTaskResponse>
```

Content REST Examples

Content operations let you export content from one CA Process Automation environment and import it into another CA Process Automation environment.

Import

You can import a local file of CA Process Automation objects.

For example, send an HTTP POST request to:

`http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/_ops/Import`

Include a request header:

`Content-Type=application/xml`

For example:

```
<ImportRequest
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:type="p1:ImportRequest"
               xmlns:p1="http://ns.ca.com/2011/09/pam-ops" >
  <ImportLocation>/TestContent/</ImportLocation>
  <SourceLocation>C:\Program
Files\CA\PAM40\standalone\.c2orepository\public\content\ITPAMConte
nt.xml</SourceLocation>
  <OverwriteAction>ImportAndReplace</OverwriteAction>
  <SetCurrent>true</SetCurrent>
</ImportRequest>
```

If the library already contains an object with the specified name, set <OverwriteAction> to one of the following options:

Import

Increase the version of the imported object if an object of the same exists in the specified import folder. Overrides the release version if the same release version exists.

DoNotImport

Do not import objects with the same name as an existing object.

ImportAndReplace

Import and replace the existing object. It deletes the previous versions of the object.

The response indicates whether the import was successful:

```
<?xml version="1.0" encoding="utf-8"?>
<pam-ops:ImportResponse
xmlns:pam-ops="http://ns.ca.com/2011/09/pam-ops"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                        xsi:type="pam-ops:ImportResponse">
  <SuccessMessage>The specified object imported
successfully.</SuccessMessage>
</pam-ops:ImportResponse>
```

Export

You can export CA Process Automation objects into a local file.

For example, send an HTTP POST request to:

```
http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/_ops/Export
```

Include a request header:

```
Content-Type=application/xml
```

For example:

```
<ExportRequest
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="p1:ExportRequest"
xmlns:p1="http://ns.ca.com/2011/09/pam-ops">
<FolderName>/rest</FolderName>
<ExportAsContentPackage>true</ExportAsContentPackage>
<IsAbsolute>true</IsAbsolute>
<ExportFileName>
<ExportLocation>C:\\REST\\ExportContent.xml</ExportLocation>
<OverwriteFile>true</OverwriteFile>
</ExportFileName>
<Filter>
<ObjectTypes>
<ObjectType><Type>Process</Type></ObjectType>
<ObjectType><Type>Agenda</Type></ObjectType>
<ObjectType><Type>ContentPackage</Type></ObjectType>
</ObjectTypes>
</Filter>
</ExportRequest>
```

Note: The SealModifiableReleaseVersions element is not required and its value is implicitly decided based on the value of ExportAsContentPackage element.

The <ExportAsContentPackage> flag specifies whether you can export a folder as a content package. The default value is false.

If you set the flag to true, CA Process Automation exports the folder as a content package.

Note: If the release version for the folder and its child objects is not defined, the export of a content package fails.

The response indicates whether the export was successful:

```
<?xml version="1.0" encoding="utf-8"?>
<pam-ops:ExportResponse
xmlns:pam-ops="http://ns.ca.com/2011/09/pam-ops"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                        xsi:type="pam-ops:ExportResponse">
  <ExportMessage>Successfully uploaded the exported library to
download file name: C:\REST\ExportContent.xml</ExportMessage>
</pam-ops:ExportResponse>
```

Encrypted XML Files

The Catalyst Import and Export web service methods support encrypted XML files. An XML file that is exported from CA Process Automation is encrypted if it contains any object with a nonmodifiable Release Version attribute.

- You can import a clear or an encrypted XML file.
- You can export a clear or an encrypted XML file. If CA Process Automation detects that the Release Version of one of the exported files is locked, it encrypts the entire XML file. If CA Process Automation detects that the Release Versions of all of the exported files are unlocked, it exports the XML file unencrypted.

Dataset REST Examples

You can query and update global named datasets. To perform the updates, use JavaScript code as you would in the pre-execution and post-execution code for a process.

QueryDatasetNames

The QueryDatasetNames function lets you query for dataset names with an optional filter on the path.

For example, send an HTTP POST request to:

```
http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/_ops/QueryDatasetNames
```

Include a request header:

```
Content-Type=application/xml
```

For example:

```
QueryDatasetNamesRequest
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                     xsi:type="p1:QueryDatasetNamesRequest"
                     xmlns:p1="http://ns.ca.com/2011/09/pam-ops"
>
      <PathName>/TestContent</PathName>
</QueryDatasetNamesRequest>
```

The response contains the path names of the datasets. The following example shows the path names in **bold**:

```
<pam-ops:QueryDatasetNamesResponse
xmlns:pam-ops="http://ns.ca.com/2011/09/pam-ops"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:type="pam-ops:QueryDatasetNamesResponse">
  <Datasets>
    <Dataset>
      <PathName>/TestContent/DatasetTest</PathName>
    </Dataset>
    <Dataset>
      <PathName>/TestContent/TestDataset</PathName>
    </Dataset>
    <Dataset>
      <PathName>/TestContent/TestDataset</PathName>
    </Dataset>
    <Dataset>
      <PathName>/TestContent/TestDataset_1</PathName>
    </Dataset>
  </Datasets>
</pam-ops:QueryDatasetNamesResponse>>
```

QueryDatasetParameters

This function searches the path that you specify between the <PathName> and </PathName> tags in the request for dataset parameters.

For example, send an HTTP POST request to:

```
http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/_ops/QueryDa
tasetParameters
```

Include a request header:

```
Content-Type=application/xml
```

For example:

```
<QueryDatasetParametersRequest
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                      xsi:type="p1:QueryDatasetParametersRequest"
                      xmlns:p1="http://ns.ca.com/2011/09/pam-ops"
>
  <PathName>/TestDatasets/TestDataset</PathName>
</QueryDatasetParametersRequest>
```

The response contains the dataset parameters, including their JavaScript names:

```
<pam-ops:QueryDatasetParametersResponse
  xmlns:pam-ops=http://ns.ca.com/2011/09/pam-ops
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:type="pam-ops:QueryDatasetParametersResponse">
  <Dataset>
    <PathName>/TestDatasets/TestDataset</PathName>
    <Parameters>
      <Param>
        <Name>Datasets["/TestDatasets/TestDataset"].Name</Name>
        <Value>John Doe</Value>
        <Label>Name</Label>
        <Type>String</Type>
      </Param>
      <Param>
        <Name>Datasets["/TestDatasets/TestDataset"].Address</Name>
        <Value>1400 Any Avenue</Value>
        <Label>Address</Label>
        <Type>String</Type>
      </Param>
      <Param>
        <Name>Datasets["/TestDatasets/TestDataset"].City</Name>
        <Value>DC</Value>
        <Label>City</Label>
        <Type>String</Type>
      </Param>
      <Param>
        <Name>Datasets["/TestDatasets/TestDataset"].Zipcode</Name>
        <Value>00000</Value>
        <Label>Zipcode</Label>
        <Type>Integer</Type>
      </Param>
      <Param>

<Name>Datasets["/TestDatasets/TestDataset"].VMap.VarString</Name>
        <Value>updated1</Value>
        <Label>VarString</Label>
        <Type>String</Type>
      </Param>
      <Param>

<Name>Datasets["/TestDatasets/TestDataset"].VMap.VarDate</Name>
        <Value xsi:nil="true"/>
        <Label>VarDate</Label>
        <Type>Date</Type>
      </Param>
      <Param>
    </Parameters>
  </Dataset>
```

```
</pam-ops:QueryDatasetParametersResponse>
```

QueryProcessDataset

This function searches for a process dataset. You specify the process instance name that includes the dataset that you want to query in the <InstanceName> tag in the REST request.

For example, send an HTTP POST request to:

```
http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/_ops/QueryProcessDataset
```

Include a request header:

```
Content-Type=application/xml
```

For example, a REST request could be:

```
<QueryProcessDatasetRequest
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                      xsi:type="p1:QueryProcessDatasetRequest"
                      xmlns:p1="http://ns.ca.com/2011/09/pam-ops"
>
  <InstanceName>Process_388</InstanceName>
</QueryProcessDatasetRequest>
```

The response returns the process dataset under the <ProcessDataset> tag. The REST response returns any parameters that are included in the dataset. For example:

```
    <?xml version="1.0" encoding="UTF-8"?>
    <pam-ops:QueryProcessDatasetResponse
xmlns:pam-ops="http://ns.ca.com/2011/09/pam-ops"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="pam-ops:QueryProcessDatasetResponse">
    <ProcessDataset>
    <InstanceName>Process_388</InstanceName>
    <Parameters>
    <Param>
    <Name>ParentProcessROID</Name>
    <Value/>
    <Label>ParentProcessROID</Label>
    <Type>String</Type>
    </Param>
    <Param>
    <Name>RootProcessROID</Name>
    <Value/>
    <Label>RootProcessROID</Label>
    <Type>String</Type>
    </Param>
    <Param>
    <Name>Progress</Name>
    <Value>100.0</Value>
    <Label>Progress</Label>
    <Type>Double</Type>
    </Param>
    <Param>
    <Name>TouchpointName</Name>
    <Value>Orchestrator</Value>
    <Label>TouchpointName</Label>
    <Type>String</Type>
    </Param>
    <Param>
    <Name>ServerName</Name>
    <Value>SINRA22.ca.com</Value>
    <Label>ServerName</Label>
    <Type>String</Type>
    </Param>
    <Param>
    <Name>ServerId</Name>
    <Value>1ed56246-7880-4c10-8a5f-5a9975c17d9b</Value>
    <Label>ServerId</Label>
    <Type>String</Type>
    </Param>
    <Param>
    <Name>DisplayName</Name>
    <Value>Process</Value>
```

```
<Label>DisplayName</Label>
<Type>String</Type>
</Param>
<Param>
<Name>InstanceUUID</Name>
<Value>024d68c3-1248-484b-aca1-ef8584492eac</Value>
<Label>InstanceUUID</Label>
<Type>String</Type>
</Param>
<Param>
<Name>rootUUID</Name>
<Value>024d68c3-1248-484b-aca1-ef8584492eac</Value>
<Label>rootUUID</Label>
<Type>String</Type>
</Param>
<Param>
<Name>CallerUser</Name>
<Value>pamadmin</Value>
<Label>CallerUser</Label>
<Type>String</Type>
</Param>
<Param>
<Name>effectiveUser</Name>
<Value>pamadmin</Value>
<Label>effectiveUser</Label>
<Type>String</Type>
</Param>
<Param>
<Name>RuntimeROID</Name>
<Value>388</Value>
<Label>RuntimeROID</Label>
<Type>Long</Type>
</Param>
<Param>
<Name>InstanceName</Name>
<Value>Process_388</Value>
<Label>InstanceName</Label>
<Type>String</Type>
</Param>
<Param>
<Name>ScheduledStartTime</Name>
<Value>01/14/2013</Value>
<Label>ScheduledStartTime</Label>
<Type>Date</Type>
</Param>
<Param>
<Name>StartDate</Name>
<Value>01/14/2013</Value>
<Label>StartDate</Label>
```

```
<Type>Date</Type>
</Param>
<Param>
<Name>StartTime</Name>
<Value>01/14/2013</Value>
<Label>StartTime</Label>
<Type>Date</Type>
</Param>
<Param>
<Name>EndTime</Name>
<Value>01/14/2013</Value>
<Label>EndTime</Label>
<Type>Date</Type>
</Param>
<Param>
<Name>EndDate</Name>
<Value>01/14/2013</Value>
<Label>EndDate</Label>
<Type>Date</Type>
</Param>
<Param>
<Name>Result</Name>
<Value>1</Value>
<Label>Result</Label>
<Type>String</Type>
</Param>
</Parameters>
</ProcessDataset>
</pam-ops:QueryProcessDatasetResponse>
```

UpdateDatasetParameters

To update dataset parameters using REST, specify the name of the parameter from the QueryDatasetParameters response. You can use expressions in the value.

For example, send an HTTP POST request to:

`http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/_ops/UpdateDatasetParameters`

Include a request header:

```
Content-Type=application/xml
```

For example:

```
<UpdateDatasetParametersRequest

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                      xsi:type="p1:UpdateDatasetParametersRequest"
xmlns:p1="http://ns.ca.com/2011/09/pam-ops"
>
  <Dataset>
    <PathName>/TestDatasets/TestDataset</PathName>
<Parameters>
      <Param>
        <Name>Datasets["/TestDatasets/TestDataset"].Zipcode</Name>
<Value>Datasets["/TestDatasets/TestDataset"].Zipcode +
100</Value>
  <Type>Integer</Type>
      </Param>
    </Parameters>
  </Dataset>
</UpdateDatasetParametersRequest>
```

The response contains the updated dataset:

```
<?xml version="1.0" encoding="UTF-8"?>
<pam-ops:UpdateDatasetParametersResponse
            xmlns:pam-ops="http://ns.ca.com/2011/09/pam-ops"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:type="pam-ops:UpdateDatasetParametersResponse">
  <Dataset>
    <PathName>/TestDatasets/TestDataset</PathName>
    <Parameters>
      <Param>
        <Name>Datasets["/TestDatasets/TestDataset"].Name</Name>
        <Value>Harry Truman</Value>
        <Label>Name</Label>
        <Type>String</Type>
      </Param>
      <Param>
        <Name>Datasets["/TestDatasets/TestDataset"].Address</Name>
        <Value>1400 Pennsylvania Avenue</Value>
        <Label>Address</Label>
        <Type>String</Type>
      </Param>
      <Param>
        <Name>Datasets["/TestDatasets/TestDataset"].City</Name>
        <Value>DC</Value>
        <Label>City</Label>
        <Type>String</Type>
      </Param>
      <Param>
        <Name>Datasets["/TestDatasets/TestDataset"].Zipcode</Name>
        <Value>2300</Value>
        <Label>Zipcode</Label>
        <Type>Integer</Type>
      </Param>
```

Note: Before updating a global dataset, make sure that the dataset has been checked in at least once.

Module Configuration REST Examples

Module configuration operations let you query and update CA Process Automation modules as you would in the Modules tab in the Configuration Browser.

QueryModuleConfigs

To query for module names, use the QueryModuleConfigs function.

For example, send an HTTP POST request to:

```
http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/_ops/QueryModuleConfigs
```

Include a request header:

```
Content-Type=application/xml
```

The following example returns all the modules:

```
<QueryModuleConfigsRequest
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                     xsi:type="p1:QueryModuleConfigsRequest"
                     xmlns:p1="http://ns.ca.com/2011/09/pam-ops" >
  <ModuleName xsi:nil="true"/>
  (Optional) <ModuleType>ALL</ModuleType>
</QueryModuleConfigsRequest>
```

The following example returns the custom modules:

```
<QueryModuleConfigsRequest
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                     xsi:type="p1:QueryModuleConfigsRequest"
                     xmlns:p1="http://ns.ca.com/2011/09/pam-ops" >
  <ModuleName xsi:nil="true"/>
  (Optional) <ModuleType>CUSTOM</ModuleType>
</QueryModuleConfigsRequest>
```

The response contains the module names:

```
<?xml version="1.0" encoding="utf-8"?>
<pam-ops:QueryModuleConfigsResponse
               xmlns:pam-ops="http://ns.ca.com/2011/09/pam-ops"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:type="pam-ops:QueryModuleConfigsResponse">
  <Module>
    <ModuleName>FTPServices</ModuleName>
    <Name>File Transfer</Name>
    <DisplayName>File Transfer</DisplayName>
    <Description>Provides file transfer operations
(FTP/SFTP).</Description>
  </Module>
  <Module>
    <ModuleName>WorkflowServices</ModuleName>
    <Name>Process Control</Name>
    <DisplayName>Process Control</DisplayName>
    <Description>Runs, monitors and controls CA Process Automation
Processes.</Description>
  </Module>
  <Module>
    <ModuleName>UtilitiesGroup</ModuleName>
    <Name>Utilities</Name>
    <DisplayName>Utilities</DisplayName>
    <Description>This module consists of utility operators which are
used in PAM processes</Description>
  </Module>
  <Module>
    <ModuleName>SOAP Services</ModuleName>
    <Name>Web Services</Name>
    <DisplayName>Web Services</DisplayName>
    <Description>Provides an interface to external services exposed
through SOAP.</Description>
  </Module>
  <Module>
    <ModuleName>JMX.Group.Name</ModuleName>
    <Name>Java Management</Name>
    <DisplayName>Java Management</DisplayName>
    <Description>Provides a management interface to external system
that support JMX.</Description>
  </Module>
  <Module>
    <ModuleName>Process Services</ModuleName>
    <Name>Command Execution</Name>
    <DisplayName>Command Execution</DisplayName>
    <Description>Runs programs and scripts on host operating
environments.</Description>
  </Module>
```

```
  <Module>
    <ModuleName>Date Time Services</ModuleName>
    <Name>Date-Time</Name>
    <DisplayName>Date-Time</DisplayName>
    <Description>Executes time and calendar constraints in CA Process
Automation processes.</Description>
  </Module>
  <Module>
    <ModuleName>FileService</ModuleName>
    <Name>File Management</Name>
    <DisplayName>File Management</DisplayName>
    <Description>This module monitors directory, files, and their
contents</Description>
  </Module>
```

QueryModuleConfigProperties

To query for the module properties, use the QueryModuleConfigProperties function.

For example, send an HTTP POST request to:

```
http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/_ops/QueryMo
duleConfigProperties
```

Include a request header:

```
Content-Type=application/xml
```

For example:

```
<QueryModuleConfigPropertiesRequest
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:type="p1:QueryModuleConfigPropertiesRequest"
                      xmlns:p1="http://ns.ca.com/2011/09/pam-ops"
>
  <ModuleName>WorkflowServices</ModuleName>
</QueryModuleConfigPropertiesRequest>
```

The response contains the module properties:

```
<?xml version="1.0" encoding="UTF-8"?>
<pam-ops:QueryModuleConfigPropertiesResponse
           xmlns:pam-ops="http://ns.ca.com/2011/09/pam-ops"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:type="pam-ops:QueryModuleConfigPropertiesResponse">
  <ModuleName>WorkflowServices</ModuleName>
  <Property>
    <Name>TimeToKeepPrompts</Name>
    <Value>5</Value>
    <Label>Time to keep completed user interactions (mins)</Label>
    <Description>This parameter determines the maximum time in
minutes that a particular prompt information remains accessible to the
user once the prompt has ended.</Description>
    <Page>Default Process Control Properties</Page>
    <Type>Long</Type>
  </Property>
</pam-ops:QueryModuleConfigPropertiesResponse>
```

UpdateModuleConfigProperties

To update the properties of a module, use the UpdateModuleConfigProperties function.

For example, send an HTTP POST request to:

```
http://<hostname>:7000/node/rest/CA:00074_CA:00074:01/_ops/UpdateM
oduleConfigProperties
```

Include a request header:

```
Content-Type=application/xml
```

For example:

```
<UpdateModuleConfigPropertiesRequest
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:type="p1:UpdateModuleConfigPropertiesRequest"
                      xmlns:p1="http://ns.ca.com/2011/09/pam-ops"
>
  <ModuleName>WorkflowServices</ModuleName>
  <Property>
    <Name>TimeToKeepPrompts</Name>
    <Value>4</Value>
    <Type>Long</Type>
  </Property>
</UpdateModuleConfigPropertiesRequest>
```

The response contains the updated module properties:

```
<?xml version="1.0" encoding="utf-8"?>
<pam-ops:UpdateModuleConfigPropertiesResponse
         xmlns:pam-ops="http://ns.ca.com/2011/09/pam-ops"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:type="pam-ops:UpdateModuleConfigPropertiesResponse">
  <Property>
    <Name>TimeToKeepPrompts</Name>
    <Value>4</Value>
    <Type>Long</Type>
  </Property>
</pam-ops:UpdateModuleConfigPropertiesResponse>
```

Catalyst OData Usage

The Open Data Protocol (OData) is a web protocol for querying and updating data. OData provides a way to unlock your data and free it from application silos. OData does this by applying and building upon web technologies such as HTTP, Atom Publishing Protocol (AtomPub), and JSON to provide access to information from various applications, services, and stores. OData exposes and accesses information from various sources including relational databases, file systems, content management systems, and traditional websites.

Note: For more information about OData, see http://www.odata.org/.

OData adds the following functionality to AtomPub:

- A convention for representing structured data
- A resource addressing scheme and URL syntax
- Common query options such as filter and sort
- A schema that describes resource structure, links, and metadata
- Payload formats and semantics for batch and unit-of-work requests
- Alternate representations of resource content (JSON)

The Catalyst OData URL has the following general format:

`http://<hostname>:7000/node/odata/<module>`

The OData URL follows the same syntax as the REST URLs, except that you specify "odata" instead of "rest". The default format is AtomPub. To retrieve JSON format, append "?$format=json" to the URL.

You can also specify OData query filters. For example, code similar to the following queries aborted processes:

```
http:/<hostname>:7000/node/odata/CA:00074_CA:00074:01/ITActivity?$
filter=StateDescription eq 'Aborted'
```

Note: For more information about the OData specification, see http://www.odata.org.

Chapter 3: SOAP API Reference

Simple Object Access Protocol (SOAP) is a protocol specification for exchanging structured information in the implementation of Web services.

The WSDL for the CA Process Automation Web services interface can be accessed at:

http(s)://<*CA Process Automation Server Name*>:< *port*>/itpam/soap?wsdl

Notes

- Incoming CA Process Automation Web services only support SOAP version 1.1.
- The Web Services methods will not be extended past CA Process Automation Release 04.1.00. The documentation is kept for reference purposes only.

This section contains the following topics:

Web Services Methods

To execute a SOAP call, you can either use a token or a combination of username and password.

The following example shows a username/password authentication and authorization:

```
CA AuthMinder
     <user>Joe</user >
     <password>thisismypassword</password >
</auth>
```

Note: See Common Tags for Web Services Methods to learn about common tags that apply to numerous Web services methods.

AsyncSoapResponse

Use this method to complete an Invoke SOAP Method Async operator that is running in any process instance.

Example

```
<tns:AsyncSoapResponse xmlns:tns="http://www.ca.com/itpam">
<tns:MessageID>514514e3-e8c8-4b1d-be42-3ee85e9d8d37</tns:MessageID>
<!--Any valid XML fragment-->
</tns:AsyncSoapResponse>
```

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<tns:AsyncSoapResponse xmlns:tns="http://www.ca.com/itpam">
<tns:MessageID>514514e3-e8c8-4b1d-be42-3ee85e9d8d37</tns:MessageID>
<!--Any valid XML fragment-->
</tns:AsyncSoapResponse></SOAP-ENV:Body></SOAP-ENV:Envelope>
```

<*MessageID*> specifies the UUID of the Invoke SOAP Method Async operator that is running.

SOAP Response

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<AsyncSoapResponseResponse xmlns="http://www.ca.com/itpam">
<status>SOAP_ASYNC_SUCCESS</status>
</AsyncSoapResponseResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

checkServerStatus

Use this method to see the status of the server.

Example

```
<tns:checkServerStatus xmlns:tns="http://www.ca.com/itpam">
  <tns:auth>
<!--xsd:Choice Type-->
<tns:token>token__</tns:token>
<tns:user>user__</tns:user>
<tns:password>password__</tns:password>
  </tns:auth>
</tns:checkServerStatus>
```

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Header/>
<SOAP-ENV:Body><tns:checkServerStatus xmlns:tns="http://www.ca.com/itpam">
  <tns:auth>
<!--xsd:Choice Type-->
<tns:token>token__</tns:token>
<tns:user>pamadmin</tns:user>
<tns:password>pamadmin</tns:password>
  </tns:auth>
</tns:checkServerStatus>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Response

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<checkServerStatusResponse xmlns="http://www.ca.com/itpam">
<serverStatus>Server status ok.</serverStatus>
</checkServerStatusResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

<*serverStatus*> shows the status of the server.

checkStartRequestStatus

Use this method to see the status of a start request.

Example

```
<tns:checkStartRequestStatus xmlns:tns="http://www.ca.com/itpam">
<tns:auth>
<!--xsd:Choice Type-->
<tns:token>token__</tns:token>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<!--xsd:Choice Type-->
<tns:interactionId>51</tns:interactionId>
<tns:tagId>tagId__</tns:tagId>
</tns:checkStartRequestStatus>
```

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<tns:checkStartRequestStatus xmlns:tns="http://www.ca.com/itpam">
<tns:auth>
<!--xsd:Choice Type-->
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:interactionId>51</tns:interactionId>
<tns:tagId>z34dsf5c7-xagb-4g5d-74g5-bd5c4we2378f</tns:tagId>
</tns:checkStartRequestStatus>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

You can include one of the following:

<*interactionId*>: The user passes the ID of the start request form to view.

<*tagId*>: The user passes the tagId of the start request form to view.

SOAP Response

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<checkStartRequestStatusResponse xmlns="http://www.ca.com/itpam">
<startRequestStatus>
<interactionId>51</interactionId>
<state>Completed</state>
<earliest-start-time>2012-03-01T12:51:25.690+05:30</earliest-start-time>
<start-time>2012-03-01T12:51:25.893+05:30</start-time>
<end-time>2012-03-01T12:51:38.033+05:30</end-time>
<refProcess>/TENSU03/Process</refProcess>
<processID>52</processID>
</startRequestStatus>
</checkStartRequestStatusResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The response contains tags under which the user can see the result.

<*state*> describes the state of the start request.

<*earliest-start-time*> states the time when the start request is queued.

<*start-time*> states the actual start time of the start request.

<*end-time*> states the end time of the start request.

<*refProcess*> shows the reference path of the process that is attached with the start request.

<*processID*> shows the Instance Id of the process (only returned if a process instance has been created).

controlInstance

Use this method to create a SOAP request to control an instance. The controlInstance method specifies archiving for the process or start request forms (and all its child instances) that are not triggered in detached mode. This method does not send a fault if a process or start request form is already marked for archiving. The controlInstance method sends a fault when a process or start request form is not found with the specified ID.

To execute a controlInstance method, use any one of the following options:

- rootUUID: This option specifies the rootUUID of the process or start request form
- instanceId: This option specifies the instance ID of the process or start request form
- tagId: This option specifies the tagid of the process or start request form.

Example

```
<tns:controlInstance xmlns:tns="http://www.ca.com/itpam">
  <tns:auth>
<!--xsd:Choice Type-->
<tns:token>token__</tns:token>
<tns:user>user__</tns:user>
<tns:password>password__</tns:password>
  </tns:auth>
  <!--xsd:Choice Type-->
  <tns:instanceId>instanceId__</tns:instanceId>
  <tns:action>archive</tns:action>
</tns:controlInstance>
```

SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:itp="http://www.ca.com/itpam">
   <soapenv:Header/>
   <soapenv:Body>
      <tns:controlInstance>
        <tns:auth>
        <!--xsd:Choice Type-->
        <tns:token>token__</tns:token>
         <tns:user>Joe</tns:user>
         <tns:password>thisismypassword</tns:password>
        <tns:auth>
        <!--xsd:Choice Type-->
        <tns:instanceId>3186</tns:instanceId>
        <tns:action>archive</tns:action>
      </tns:controlInstance>
   </soapenv:Body>
</soapenv:Envelope>
```

<*rootUUID*> specifies the rootUUID of the process/start request form.

<*instanceId*> specifies the instance ID of the process/start request form.

<*tagId*> specifies the tagid of the process/start request form.

<*action*> specifies the action to perform on the process/start request form instance. Currently, only archive is supported.

SOAP Response

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Header/>
   <SOAP-ENV:Body>
      <controlProcessResponse xmlns="http://www.ca.com/itpam">
         <actionStatus>The archive action was successfully executed.</actionStatus>
      </controlProcessResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

controlProcess

Use this method to create a request to control a process.

Example

```
<tns:controlProcess xmlns:tns="http://www.ca.com/itpam">
  <tns:ProcessID>ProcessID__</tns:ProcessID>
  <tns:action>action__</tns:action>
  <tns:auth>
<!--xsd:Choice Type-->
<tns:token>token__</tns:token>
<tns:user>user__</tns:user>
<tns:password>password__</tns:password>
  </tns:auth>
</tns:controlProcess>
```

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<SOAP-ENV:Header/>

<SOAP-ENV:Body>

<tns:controlProcess xmlns:tns="http://www.ca.com/itpam">
<tns:ProcessID>47</tns:ProcessID>
<tns:action>suspend</tns:action>
<tns:auth>
<!--xsd:Choice Type-->
<tns:token>token__</tns:token>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
</tns:controlProcess></SOAP-ENV:Body></SOAP-ENV:Envelope>
```

SOAP Response

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<controlProcessResponse xmlns="http://www.ca.com/itpam">
<actionStatus>The suspend action for Process ID "47" was queued.</actionStatus>
</controlProcessResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Actions

Suspend

You can suspend a single process instance through a web services method.

If you cannot suspend a process instance (for example, it already completed or aborted), an error message returns containing details about the request.

Here is an example of a fault when you try to suspend a completed process:

```
<Fault xmlns="http://schemas.xmlsoap.org/soap/envelope/">
<faultcode>SOAP-ENV:Server</faultcode>
<faultstring>The process with Process ID "208" is in completed state and
therefore cannot be suspended.</faultstring>
</Fault>
```

Resume

You can resume a single process instance that was previously suspended through a web services method.

If you cannot resume a process instance, an error message returns containing details about the request. For example, a process instance cannot resume because it was already completed, aborted, or not currently suspended.

Here is an example of a fault when you try to resume a completed process:

```
<Fault xmlns="http://schemas.xmlsoap.org/soap/envelope/">
<faultcode>SOAP-ENV:Server</faultcode>
<faultstring>The process with Process ID "208" is in completed state and
therefore cannot be resumed.</faultstring>
</Fault>
```

Abort

You can abort a single process instance through a web services method.

If you cannot abort a process instance (for example, it already completed or aborted), an error message returns containing details about the request.

Here is an example of a fault when you try to abort a completed process:

```
<Fault xmlns="http://schemas.xmlsoap.org/soap/envelope/">
<faultcode>SOAP-ENV:Server</faultcode>
<faultstring>The process with Process ID "486" is in completed  state and
therefore cannot be Aborted.</faultstring>
</Fault>
```

Example 1

```
<tns:controlProcess xmlns:tns="http://www.ca.com/itpam">
        <tns:ProcessID>ProcessID </tns:ProcessID>
        <tns:action>action</tns:action>
          <tns:auth>
           <!--xsd:Choice Type-->
           <tns:token>token__</tns:token>
           <tns:user>Joe</tns:user>
           <tns:password>thisismypassword</tns:password>
          <tns:auth></tns:controlProcess>
```

"*ProcessedID*" is the ROID of the process instance that you must control (suspend/ resume/abort). The "ROID" of a running process is in the response when executing a process through web services.

Example 2

```
<tns:ExecuteC2OFlowResponse xmlns:tns="http://www.ca.com/itpam">
        <tns:ExecuteC2OFlowResult>
        <tns:ROID>567</tns:ROID>
        ...
        ...
        </tns:ExecuteC2OFlowResult>
</tns:ExecuteC2OFlowResponse>
```

The value for action is to suspend, resume, or abort. Only an authorized user can perform this action; otherwise, a fault is returned.

Example 3

```
<SOAP-ENV:Fault xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
        <faultcode>SOAP-ENV:Server</faultcode>
        <faultstring>Invalid user/password or token.</faultstring>
</SOAP-ENV:Fault>
```

A fault returns if you try to perform an operation that you cannot perform on a process in its current state. For example, trying to suspend a process that already completed.

Example 4

```
<SOAP-ENV:Fault xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
        <faultcode>SOAP-ENV:Server</faultcode>
        <faultstring>Process with Process Id "576" is in completed state therefore
cannot be suspended.</faultstring>
</SOAP-ENV:Fault>
```

deleteArchivedInstances

Use this method to delete the archived instances of a CA Process Automation Orchestrator.

Example

```
<tns:deleteArchivedInstances xmlns:tns="http://www.ca.com/itpam">
<tns:auth>
<!--xsd:Choice Type-->
<tns:token>token__</tns:token>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:dateRange>
<tns:fromDate>2012-02-02T02:00:00.320+05:30</tns:fromDate>
<tns:toDate>2012-02-02T03:00:00.320+05:30</tns:toDate>
</tns:dateRange>
</tns:deleteArchivedInstances>
```

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Header/><SOAP-ENV:Body>
<tns:deleteArchivedInstances xmlns:tns="http://www.ca.com/itpam">
<tns:auth>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:dateRange>
<tns:fromDate>2012-02-02T02:00:00.320+05:30</tns:fromDate>
<tns:toDate>2012-02-02T03:00:00.320+05:30</tns:toDate>
</tns:dateRange>
</tns:deleteArchivedInstances>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

You can specify the <*dateRange*> in the <*fromDate*> and <*toDate*> tags to delete archived instances. Use the Standard XSD format 2002-05-30T09:00:00.

SOAP Response

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<deleteArchivedInstancesResponse xmlns="http://www.ca.com/itpam">
<successMessage>4 archived instances were deleted successfully.</successMessage>
</deleteArchivedInstancesResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

<*successMessage*> shows the successful deletion of instances and the number of instances deleted.

When there are no instances in the given period, the response will be:

```
<deleteArchivedInstancesResponse xmlns="http://www.ca.com/itpam">
<successMessage>No archived instance is available for the specified date
range.</successMessage>
</deleteArchivedInstancesResponse>
```

deleteAttachments

Use this method to delete the attachments which were uploaded using the executeStartRequest/ExecuteProcess method.

Example

```
<tns:deleteAttachments xmlns:tns="http://www.ca.com/itpam">
<tns:auth>
<!--xsd:Choice Type-->
<tns:token>token__</tns:token>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:attachments>
<tns:attachmentId>111</tns:attachmentId>
</tns:attachments>
</tns:deleteAttachments>
```

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<tns:deleteAttachments xmlns:tns="http://www.ca.com/itpam">
<tns:auth>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:attachments>
<tns:attachmentId>111</tns:attachmentId>
</tns:attachments>
</tns:deleteAttachments>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Response

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<deleteAttachmentsResponse xmlns="http://www.ca.com/itpam">
<successMessage>The specified attachments were deleted
successfully.</successMessage>
</deleteAttachmentsResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The <*successMessage*> tag shows results of the request.

executePendingInteraction

Use this method to execute the pending user interactions.

Example

```
<tns:executePendingInteraction xmlns:tns="http://www.ca.com/itpam">
<tns:auth>
<!--xsd:Choice Type-->
<tns:token>token__</tns:token>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:params>
<tns:param name="Var_0">hello</tns:param>
<tns:param name="Var_1">world</tns:param>
</tns:params>
<tns:userInteractionID>22141</tns:userInteractionID>
<tns:isApprove>true</tns:isApprove>
</tns:executePendingInteraction>
```

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<tns:executePendingInteraction xmlns:tns="http://www.ca.com/itpam">
<tns:auth>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:params>
<tns:param name="Var_0">hello</tns:param>
<tns:param name="Var_1">world</tns:param>
</tns:params>
<tns:userInteractionID>22141</tns:userInteractionID>
<tns:isApprove>true</tns:isApprove>
</tns:executePendingInteraction>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

To provide a parameter

In the previous request, the user can provide a list of parameters under the <params> tag. The user must create one <param> tag which includes the following parameters:

Name attribute of this tag

Provide the name of the parameter.

Value of this tag

Provide the value of the parameter.

<*userInteractionID*> is the ID of the Task Id of the user interaction execute.

<*isApprove*> specifies to approve or reject the task. This value can be true or false. This tag is optional.

SOAP Response

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<executePendingInteractionResponse xmlns="http://www.ca.com/itpam">
<userInteractionID>22141</userInteractionID>
<interactionTitle>Task_MyTasks</interactionTitle>
<state>Completed</state>
<startTime>2012-02-29T18:11:28.817+05:30</startTime>
<endTime>2012-02-29T18:17:27.707+05:30</endTime>
<parmitedUserGroup>pamadmin</parmitedUserGroup>
<closedBy>pamadmin</closedBy>
</executePendingInteractionResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

<*userInteractionID*> specifies the Id of the interaction request.

<*interactionTitle*> specifies the title of the interaction request.

<*state*> specifies the state of the interaction request.

<*startTime*> specifies the start time of the interaction request.

<*endTime*> specifies the end time of the interaction request.

<*parmitedUserGroup*> shows the list of assigned users and groups.

<*closedBy*> shows the name of the user who responded to the task.

Note: The "IsApprove flag" was being ignored for this method in CA Process Automation r2.1. To correct this issue, yet allow retention of the existing behavior for existing CA Process Automation instances, a new install has "oasis.reject.unnecessary.approval" set to true in the OasisConfig.properties configuration file. Existing installations have oasis.reject.unnecessary.approval added to set to the OasisConfig.properties configuration file and set to false.

If "oasis.reject.unnecessary.approval" is true, then the server considers the isApprove flag. A fault is returned if the user tries to approve or reject a form which does not require approval. A fault is also returned if the SOAP message does not include the isApprove flag for a form which requires approval.

executeProcess

Use this method to execute a process.

Example

```
<tns:executeProcess xmlns:tns="http://www.ca.com/itpam">
<tns:flow>
<tns:name>/MIMETEST</tns:name>
<tns:action>start</tns:action>
<tns:auth>
<!--xsd:Choice Type-->
<tns:token>token__</tns:token>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:params>
<tns:param name="name_6__">param__</tns:param>
</tns:params>
<tns:options>
<tns:startDate>startDate__</tns:startDate>
<tns:startTime>startTime__</tns:startTime>
<tns:tagId> 06a4d113-0333-4aba-8cce-8781c18647c9</tns:tagId>
<tns:isAutoArchive>true</tns:isAutoArchive>
<tns:priority>priority__</tns:priority>
</tns:options>
<tns:attachments attachmentsParamName="name">
<tns:attachment>
<tns:attachmentID>123</tns:attachmentID>
<tns:name>MIMETEST</tns:name>
<tns:localSourceLocation>"C:\\setupdir.log"</tns:localSourceLocation>
</tns:attachment>
</tns:attachments>
</tns:flow>
</tns:executeProcess>
```

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<tns:executeProcess xmlns:tns="http://www.ca.com/itpam">
<tns:flow>
<tns:name>/MIMETEST</tns:name>
<tns:action>start</tns:action>
<tns:auth>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:params>
<tns:param name="name_6__">param__</tns:param>
</tns:params>
<tns:options>
<tns:startDate>startDate__</tns:startDate>
<tns:startTime>startTime__</tns:startTime>
<tns:tagId> 06a4d113-0333-4aba-8cce-8781c18647c9</tns:tagId>
<tns:isAutoArchive>true</tns:isAutoArchive>
<tns:priority>priority__</tns:priority>
</tns:options>
<tns:attachments attachmentsParamName="TENSU03">
<tns:attachment>
<tns:attachmentID>123</tns:attachmentID>
<tns:name>MIMETEST</tns:name>
<tns:localSourceLocation>"C:\\setupdir.log"</tns:localSourceLocation>
</tns:attachment>
</tns:attachments>
</tns:flow>
</tns:executeProcess>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The following tags are under the <flow> tag:

- <*name*> specifies the name of the process including absolute path which the user wants to execute.
- <*action*> Specifies "start" to start the process.
- <*tagId*> Adds a unique ID to a process/start request form. A fault returns if there is a process or start request form instance with same tag. The <*tagId*> tag monitors the status of any process/start request form that you specify the tag for.
- <*IsAutoArchive*> tag can be set to false to skip process/Start Request Form instances (along with their child instances) from archiving under archival policy. Processes that are run in detached mode are not affected by this tag. The default value for <*IsAutoArchive*> is true.

SOAP Response

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<ExecuteC2OFlowResponse xmlns="http://www.ca.com/itpam">
<ExecuteC2OFlowResult>
<ROID>150</ROID>
<flow-name>/MIMETEST</flow-name>
<flow-action>start</flow-action>
<auth-user>pamadmin</auth-user>
<submission-time>2012-01-18T09:26:36.453+05:30</submission-time>
<sender-address>none</sender-address>
<connection-id>none</connection-id>
<message>Document accepted for processing</message>
<params>
<param name="RuntimeROID">150</param>
<param name="ServerId">1b4bda3d-bc5c-4a89-a2ad-23e2d4e71bce</param>
<param name="effectiveUser">pamadmin</param>
<param name="InstanceUUID">659322db-deca-4827-b622-14f69d09b33c</param>
<param name="name_6__">param__</param>
<param name="UserName">pamadmin</param>
<param name="ProcessName">/MIMETEST</param>
<param name="ServerName">lodivsa204.ca.com</param>
<param name="DisplayName">MIMETEST</param>
<param name="ProcessPriority">priority__</param>
<param name="TouchpointName">Orchestrator</param>
<param name="CallerUser">pamadmin</param>
<param name="ScheduledStartTime">2012-01-18 00:00:00</param>
<param name="rootUUID">659322db-deca-4827-b622-14f69d09b33c</param>
<param name="ProcessAction">start</param>
<param name="InstanceName">MIMETEST_150</param>
</params>
</ExecuteC2OFlowResult>
</ExecuteC2OFlowResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

In this response, the user can see the details of the instance of the process that executes.

<*ROID*> specifies the ROID of the instance of the process.

executeStartRequest

Use this method to start a start request form from the CA Process Automation library.

Example

```
<tns:executeStartRequest xmlns:tns="http://www.ca.com/itpam">
<tns:auth>
<!--xsd:Choice Type-->
<tns:token>token__</tns:token>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:objLocation>
<tns:name>MYSRF</tns:name>
<tns:path>/myfolder/</tns:path>
</tns:objLocation>
<tns:params>
<tns:param name="Var_0">joe</tns:param>
<tns:param name="Var_1">smith</tns:param>
</tns:params>
<tns:options>
<tns:startDate>startDate__</tns:startDate>
<tns:startTime>startTime__</tns:startTime>
<tns:tagId> 06a4d113-0333-4aba-8cce-8781c18647c9</tns:tagId>
<tns:isAutoArchive>true</tns:isAutoArchive>
<tns:priority>priority__</tns:priority>
</tns:options>
<tns:attachments attachmentsParamName="attachmentsParamName__">
<tns:attachment>
<tns:attachmentID>attachmentID__</tns:attachmentID>
<tns:name>name_11__</tns:name>
<tns:localSourceLocation>localSourceLocation__</tns:localSourceLocation>
</tns:attachment>
</tns:attachments>
</tns:executeStartRequest>
```

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<tns:executeStartRequest xmlns:tns="http://www.ca.com/itpam">
<tns:auth>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:objLocation>
<tns:name>MYSRF</tns:name>
<tns:path>/TENSU03/</tns:path>
</tns:objLocation>
<tns:params>
<tns:param name="Var_0">user</tns:param>
<tns:param name="Var_1">joe</tns:param>
</tns:params>
<tns:options>
<tns:startDate>startDate__</tns:startDate>
<tns:startTime>startTime__</tns:startTime>
<tns:tagId>06a4d113-0333-4aba-8cce-8781c18647c9</tns:tagId>
<tns:isAutoArchive>true</tns:isAutoArchive>
<tns:priority>priority__</tns:priority>
</tns:options>
<tns:attachments attachmentsParamName="attachmentsParamName__">
<tns:attachment>
<tns:attachmentID>attachmentID__</tns:attachmentID>
<tns:name>name_11__</tns:name>
<tns:localSourceLocation>localSourceLocation__</tns:localSourceLocation>
</tns:attachment>
</tns:attachments>
</tns:executeStartRequest>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

<*objLocation*> specifies the location of the start request form in the CA Process Automation library. Two tags are under the <objLocation> tag.

<*name*> specifies the name of the start request form.

<*path*> provides the absolute path of the start request form object in the CA Process Automation Library.

<*tagId*> Adds a unique ID to a process/Start Request Form. A fault returns if there is a process or start request form instance with same tag. The <*tagId*> tag monitors the status of any process/start request form that you specify the tag for.

<*IsAutoArchive*> tag can be set to false to skip process/start request form instances (along with their child instances) from archiving under archival policy. Processes that are run in detached mode are not affected by this tag. The default value for <*IsAutoArchive*> is true.

SOAP Response

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<executeStartRequestResponse xmlns="http://www.ca.com/itpam">
<startRequestStatus>
<interactionId>114</interactionId>
<state>Running</state>
<earliest-start-time>2012-03-01T00:00:00.000+05:30</earliest-start-time>
<start-time>2012-03-01T13:54:13.523+05:30</start-time>
<refProcess>/TENSU03/Process</refProcess>
<params>
<param name="DisableSchedulingDialog">True</param>
<param name="FLowChartPath">/TENSU03/Process</param>
<param name="userId">pamadmin</param>
<param name="CALLER_TYPE">ITPAM_WS</param>
<param name="Var_1">tentu</param>
<param name="Var_0">sudhakar</param>
</params>
</startRequestStatus>
</executeStartRequestResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

<*interactionId*> specifies the interaction Id of the instance of start request form started.

<*state*> shows the state of the instance.

<*refProcess*> is the name of the process with the absolute path attached with the start request form.

exportObject

Use this method to export a library using a Web service call.

Note: Ensure that the maximum XML size that you want to export is 250 MB.

Example

```
<tns:exportObject xmlns:tns="http://www.ca.com/itpam">
<tns:auth>
<!--xsd:Choice Type-->
<tns:token>token__</tns:token>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:folderName>/SOAPTest/ObjectsToExport</tns:folderName>
<tns:level>2</tns:level>
<tns:isAbsolute>true</tns:isAbsolute>
<tns:downloadFileName>
<tns:downloadLocation>C:\</tns:downloadLocation>
<tns:fileName>SOAPExportedObjects.xml</tns:fileName>
<tns:overwriteFile>true</tns:overwriteFile>
</tns:downloadFileName>
<tns:filter>
<tns:exportObjectTypes>
<tns:objectType>Agenda</tns:objectType>
<tns:objectType>Calendar</tns:objectType>
<tns:objectType>CustomIcon</tns:objectType>
<tns:objectType>CustomOperator</tns:objectType>
<tns:objectType>Dataset</tns:objectType>
<tns:objectType>InteractionRequestForm</tns:objectType>
<tns:objectType>Package</tns:objectType>
<tns:objectType>Process</tns:objectType>
<tns:objectType>ProcessWatch</tns:objectType>
<tns:objectType>Resources</tns:objectType>
<tns:objectType>StartRequestForm</tns:objectType>
</tns:exportObjectTypes>
</tns:filter>
</tns:exportObject>
```

SOAP Request

The request contains:

- FolderName (the folder or object that you want to export)
- Folder recursion level
- Absolute/relative paths in export output
- Object types
- Download file name and the download location, where the exported library has to save.

Note: All parameters are optional with the exception of FolderName.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<tns:exportObject xmlns:tns="http://www.ca.com/itpam">
<tns:auth>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:folderName>/SOAPTest/ObjectsToExport</tns:folderName>
<tns:level>2</tns:level>
<tns:isAbsolute>true</tns:isAbsolute>
<tns:downloadFileName>
<tns:downloadLocation>C:\</tns:downloadLocation>
<tns:fileName>SOAPExportedObjects.xml</tns:fileName>
<tns:overwriteFile>true</tns:overwriteFile>
</tns:downloadFileName>
<tns:filter>
<tns:exportObjectTypes>
<tns:objectType>Agenda</tns:objectType>
<tns:objectType>Calendar</tns:objectType>
<tns:objectType>CustomIcon</tns:objectType>
<tns:objectType>CustomOperator</tns:objectType>
<tns:objectType>CustomSensor</tns:objectType>
<tns:objectType>Dataset</tns:objectType>
<tns:objectType>InteractionRequestForm</tns:objectType>
<tns:objectType>Package</tns:objectType>
<tns:objectType>Process</tns:objectType>
<tns:objectType>ProcessWatch</tns:objectType>
<tns:objectType>Resources</tns:objectType>
<tns:objectType>StartRequestForm</tns:objectType>
<tns:objectType>StatePolicy</tns:objectType>
<tns:objectType>System</tns:objectType>
</tns:exportObjectTypes>
</tns:filter>
</tns:exportObject>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Response

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<ExportLibraryResponse xmlns="http://www.ca.com/itpam">
<exportLibraryResponse>Successfully uploaded the exported library to provided
download file name:C:\SOAPExportedObjects.xml</exportLibraryResponse>
<downloadedServer>PAM40-W2K8-17</downloadedServer>
</ExportLibraryResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

- If you do not provide the Download File Name in the request, then the serialized library (exported library object) sends it as an attachment to the response.
- If you do not provide an accessible shared location and file name in the request, then the error response is returned to the user.
- If you provide the accessible shared location and the Download File Name, then the exported library is saved at the given location with the specified file name. The shared location is returned in response.
- If a Download File Name that you provide exists in the shared location, then the error response is returned to the user.

generateEvent

Use this method to post an event.

Example

```
<tns:generateEvent xmlns:tns="http://www.ca.com/itpam">
<tns:auth>
<!--xsd:Choice Type-->
<tns:token>token__</tns:token>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:eventName>test</tns:eventName>
<tns:eventType>eventType__</tns:eventType>
<tns:eventSource>testsource</tns:eventSource>
<tns:eventDestination>testdestination</tns:eventDestination>
<tns:eventExpDuration>130</tns:eventExpDuration>
<!--xsd:Choice Type-->
<tns:payload>test</tns:payload>
<tns:params>
<tns:param name="name__">param__</tns:param>
</tns:params>
<tns:toSingleSubscriber>false</tns:toSingleSubscriber>
</tns:generateEvent>
```

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<SOAP-ENV:Header/>

<SOAP-ENV:Body>

<tns:generateEvent xmlns:tns="http://www.ca.com/itpam">
<tns:auth>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:eventName>test</tns:eventName>
<tns:eventType>eventType__</tns:eventType>
<tns:eventSource>testsource</tns:eventSource>
<tns:eventDestination>testdestination</tns:eventDestination>
<tns:eventExpDuration>130</tns:eventExpDuration>
<tns:payload>test</tns:payload>
<tns:params>
<tns:param name="name__">param__</tns:param>
</tns:params>
<tns:toSingleSubscriber>false</tns:toSingleSubscriber>
</tns:generateEvent>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

<*eventName*> identifies the name of the event (mandatory).

<*eventType*> identifies the type of event (optional).

<*eventSource*> identifies the source of the event (optional).

<*eventDestination*> identifies the destination of the event (optional).

<*eventExpDuration*> identifies the expiration duration of the event.

<*payload*> and <*params*> identify additional event data. If it is a single value, use payload; otherwise, use params.

<*toSingleSubscriber*> identifies whether the event can be delivered to single or multiple subscribers.

SOAP Response

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<generateEventResponse xmlns="http://www.ca.com/itpam">
<event>
<eventId>b1a3c218-0a7d-4667-9b7c-9eeddcbc7408</eventId>
<eventName>test</eventName>
<eventType>eventType__</eventType>
<eventSource>testsource</eventSource>
<eventDestination>testdestination</eventDestination>
<payload>test</payload>
<toSingleSubscriber>false</toSingleSubscriber>
<eventCreationTime>2012-03-01T14:05:14.692+05:30</eventCreationTime>
<eventExpirationTime>2012-03-01T14:07:24.692+05:30</eventExpirationTime>
<user>pamadmin</user>
</event>
</generateEventResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

getAttachments

Use this method to view attachments.

Example

```
<tns:getAttachments xmlns:tns="http://www.ca.com/itpam">
<tns:auth>
<!--xsd:Choice Type-->
<tns:token>token__</tns:token>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:filter>
<tns:contentId>111</tns:contentId>
<tns:contentType>text/html</tns:contentType>
</tns:filter>
</tns:getAttachments>
```

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Header/>
<SOAP-ENV:Body><tns:getAttachments xmlns:tns="http://www.ca.com/itpam">
<tns:auth>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:filter>
<tns:contentId>111</tns:contentId>
<tns:contentType>text/html</tns:contentType>
</tns:filter>

</getAttachments></SOAP-ENV:Body></SOAP-ENV:Envelope>
```

<*filter*> tag filters the response that is based on the content ID and content type of the attachment (optional).

<*contentId*> specifies the content ID of the attachment (optional).

<*contentType*> specifies the content Type of the attachment (optional).

SOAP Response

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<getAttachmentsResponse xmlns="http://www.ca.com/itpam">
<attachments>
<attachment>
<attachmentID>1</attachmentID>
<name>test</name>
<contentId>111</contentId>
<contentType>text/html</contentType>
</attachment>
</attachments>
</getAttachmentsResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

In the response for every attachment found, there is an <*attachment*> tag.

For each attachment tag, there are the following tags:

<*attachmentID*> specifies the Attachment ID of the attachment.

<*name*> specifies the name of the attachment.

<*contentId*> specifies the content ID of the attachment.

<*contentType*> specifies the content type of the attachment.

getITPamVersionInfo

Use this method to obtain CA Process Automation version details.

Example

```
<tns:getITPamVersion xmlns:tns="http://www.ca.com/itpam">
<tns:auth>
<!--xsd:Choice Type-->
<tns:token>token__</tns:token>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
</tns:getITPamVersion>
```

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<tns:getITPamVersion xmlns:tns="http://www.ca.com/itpam">
<tns:auth>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
</tns:getITPamVersion>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

<*servicePack*> identifies the current CA Process Automation service pack (optional).

<*patchNumber*> identifies the current CA Process Automation patch number (optional).

<*buildDetails*> identifies any build details being used with the current CA Process Automation version (optional).

<*majorVersion*> identifies the current CA Process Automation version.

<*minorVersion*> identifies the current CA Process Automation minor version.

<*buildNumber*> identifies the current CA Process Automation build number.

<*buildDate*> identifies the current CA Process Automation build date.

SOAP Response

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<ITPamVersionDetails xmlns="http://www.ca.com/itpam">
        <majorVersion>4</majorVersion>
        <minorVersion>0</minorVersion>
        <patchNumber>0</patchNumber>
        <buildNumber>335</buildNumber>
        <buildDate>2012-03-01T06:08:09.000+05:30</buildDate>
</ITPamVersionDetails>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

<*majorVersion*> identifies the current CA Process Automation version.

<*minorVersion*> identifies the current CA Process Automation minor version.

<*patchNumber*> identifies the current CA Process Automation patch number (optional).

<*buildNumber*> identifies the current CA Process Automation build number.

<*buildDate*> identifies the current CA Process Automation build date.

<*buildDetails*> provides additional details about the build (optional).

getMatchingEvents

Use this method to obtain the list of matching events for a given subscriber.

Example

Allow partial Match True

```
<tns:getMatchingEvents xmlns:tns="http://www.ca.com/itpam">
<tns:auth>
<!--xsd:Choice Type-->
<tns:token>token__</tns:token>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:eventName>abc</tns:eventName>
<tns:eventType></tns:eventType>
<tns:eventSource></tns:eventSource>
<tns:eventDestination></tns:eventDestination>
<tns:eventExpression></tns:eventExpression>
<tns:enablePatternMatch></tns:enablePatternMatch>
<tns:allowPartialMatch>true</tns:allowPartialMatch>
<tns:consumeEvents></tns:consumeEvents>
<tns:retrieveAllMatchingEvents>true</tns:retrieveAllMatchingEvents>
<tns:uniqueId></tns:uniqueId>
</tns:getMatchingEvents>
```

Enable Pattern Match True

```
<tns:getMatchingEvents xmlns:tns="http://www.ca.com/itpam">
<tns:auth>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:eventName>a.*f</tns:eventName>
<tns:eventType></tns:eventType>
<tns:eventSource></tns:eventSource>
<tns:eventDestination></tns:eventDestination>
<tns:eventExpression></tns:eventExpression>
<tns:enablePatternMatch>true</tns:enablePatternMatch>
<tns:allowPartialMatch></tns:allowPartialMatch>
<tns:consumeEvents></tns:consumeEvents>
<tns:retrieveAllMatchingEvents>true</tns:retrieveAllMatchingEvents>
<tns:uniqueId></tns:uniqueId>
</tns:getMatchingEvents>
```

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<tns:getMatchingEvents xmlns:tns="http://www.ca.com/itpam">
<tns:auth>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:eventName>abc</tns:eventName>
<tns:eventType/>
<tns:eventSource/>
<tns:eventDestination/>
<tns:eventExpression/>
<tns:enablePatternMatch/>
<tns:allowPartialMatch/>
<tns:consumeEvents/>
<tns:retrieveAllMatchingEvents>true</tns:retrieveAllMatchingEvents>
<tns:uniqueId/>
</tns:getMatchingEvents>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

<*eventName*> identifies the name of the event (optional).

<*eventType*> identifies the type of event (optional).

<*eventSource*> identifies the source of the event (optional).

<*eventDestination*> identifies the destination of the event (optional).

<*eventExpression*> is a Boolean expression for additional event parameters. This expression is evaluated against the payload or event parameters that are sent with an event (optional). For example:

```
Event.eventid==1234.
```

<*enablePatternMatch*> enables pattern matching for the parameters like event name, type, source and destination.

<*allowPartialMatch*> allows a partial match for the parameters like event name, type, source and destination.

<*uniqueId*> sends a unique ID with the request so that the SOAP call consumes the event only once. If the send event has an option to deliver to multiple subscribers, multiple calls of getMatchingEvents with same <*uniqueId*> consumes the event only once.

<*consumeEvents*> lets the request consume or not consume the matching events.

<*retrieveAllMatchingEvents*> lets the request retrieve all or first matched send event.

SOAP Response

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<getMatchingEventsResponse xmlns="http://www.ca.com/itpam">
<events>
<event>
<eventId>fefce4c1-ad8e-42a2-8964-362704ddbb24</eventId>
<eventName>abc</eventName>
<eventType>eventType__</eventType>
<eventSource>eventSource__</eventSource>
<eventDestination>eventDestination__</eventDestination>
<payload>payload__</payload>
<toSingleSubscriber>false</toSingleSubscriber>
<eventCreationTime>2012-03-06T17:04:32.562+05:30</eventCreationTime>
<eventExpirationTime>2012-03-06T17:06:42.562+05:30</eventExpirationTime>
<user>pamadmin</user>
</event>
</events>
</getMatchingEventsResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

getPendingInteractionRequestForm

Use this method to retrieve information for a pending interaction request form that is based on the Task ID.

Example

```
<tns:getPendingInteractionRequestForm xmlns:tns="http://www.ca.com/itpam"
getApprovalRequired="false" getParamSequence="false">
<tns:auth>
<!--xsd:Choice Type-->
<tns:token>token__</tns:token>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:userInteractionID>924</tns:userInteractionID>
</tns:getPendingInteractionRequestForm>
```

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<tns:getPendingInteractionRequestForm xmlns:tns="http://www.ca.com/itpam"
getApprovalRequired="false" getParamSequence="false">
<tns:auth>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:userInteractionID>924</tns:userInteractionID>
</tns:getPendingInteractionRequestForm>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

<*userInteractionID*> specifies the Task ID of the pending interaction request form.

<*getPendingInteractionRequestForm*> retrieves the sequence number of the parameters and sets the getParamSequence attribute value to "true".

SOAP Response

With proper values:

```
<getPendingInteractionRequestFormResponse xmlns="http://www.ca.com/itpam">
<processID>913</processID>
<isApprovalRequired>true</isApprovalRequired>
<description/>
<pages hasComplexType="false">
<page name="Parameters">
<itpamString isReadOnly="false" maxlength="2147483647" minlength="0" name="Var_0"
sequenceNo="0">
<label>Var_0</label>
<description/>
<value/>
</itpamString>
<itpamString isReadOnly="false" maxlength="2147483647" minlength="0" name="Var_1"
sequenceNo="1">
<label>Var_1</label>
<description/>
<value/>
</itpamString>
</page>
<page name="System"/>
<page name="User Prompt"/>
</pages>
</getPendingInteractionRequestFormResponse>
```

<*pages*> specifies the details of the pages in the interaction request form.

When provided a task in taken state:

```
<getPendingInteractionRequestFormResponse xmlns="http://www.ca.com/itpam">
<processID>7</processID>
<isApprovalRequired>true</isApprovalRequired>
<description/>
<pages hasComplexType="false">
<page name="Parameters">
<itpamString isReadOnly="false" maxlength="2147483647" minlength="0" name="Var_0"
sequenceNo="0">
<label>Var_0</label>
<description/>
<value/>
</itpamString>
<itpamString isReadOnly="false" maxlength="2147483647" minlength="0" name="Var_1"
sequenceNo="1">
<label>Var_1</label>
<description/>
<value/>
</itpamString>
</page>
<page name="System"/>
<page name="User Prompt"/>
</pages>
</getPendingInteractionRequestFormResponse>
```

When provided a task in completed state:

```
<Fault xmlns="http://schemas.xmlsoap.org/soap/envelope/">
<faultcode>SOAP-ENV:Server</faultcode>
<faultstring>"UserInteraction with Task Id "<<taskid>>" is in completed state
therefore this action is not allowed"
</faultstring>
</Fault>
```

Note: Starting with CA Process Automation r2.2, "getApprovalRequired" is added to "getPendingInteractionRequestForm" request. If it is set to true, then the "isApprovalRequired" flag returns, indicating if the form needs approval. To enable this behavior in CA Process Automation instances that were upgraded from r2.1, have your CA Process Automation Administrator perform the changes to the CA Process Automation configuration (described in the Note in "executePendingInformation Method (see page 116)").

getPendingUserInteractions

Use this method to view pending user interactions. The number of pending interactions that return is based on your user role:

- If the user is an administrator, all user interactions that are pending in the Orchestrator return.
- For any other role, only those pending interactions that are assigned to the specific user return.

Return all interactions that are pending in the Orchestrator (Administrators only)

Inline text

```
<tns:getPendingUserInteractions xmlns:tns="http://www.ca.com/itpam"
getRootUUID="false">
<tns:auth>
<!--xsd:Choice Type-->
<tns:token>token__</tns:token>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:rootUUID></tns:rootUUID>
<tns:processID></tns:processID>
</tns:getPendingUserInteractions>
```

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<tns:getPendingUserInteractions xmlns:tns="http://www.ca.com/itpam"
getRootUUID="false">
<tns:auth>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:rootUUID></tns:rootUUID>
<tns:processID></tns:processID>
</tns:getPendingUserInteractions>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response

```
<getPendingUserInteractionsResponse xmlns="http://www.ca.com/itpam">
<userInteractions>
<userInteraction id="116" title="Task by B">
<description>Task by B</description>
</userInteraction>
<userInteraction id="117" title="Task by B">
<description>Task by B</description>
</userInteraction>
<userInteraction id="118" title="Task by B">
<description>Task by B</description>
</userInteraction>
<userInteraction id="119" title="Task by B">
<description>Task by B</description>
</userInteraction>
<userInteraction id="120" title="Task by B">
<description>Task by B</description>
</userInteraction>
<userInteraction id="121" title="Task by B">
<description>Task by B</description>
</userInteraction>
```

Return all pending interactions that are invoked from a start request form

Inline text

Provide the rootUUID of the process that the start request form is calling:

```
<tns:getPendingUserInteractions xmlns:tns="http://www.ca.com/itpam"
getRootUUID="false">
<tns:auth>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:rootUUID>076be822-7514-4f04-a333-3e8b73693a2b</tns:rootUUID>
<tns:processID></tns:processID>
</tns:getPendingUserInteractions>
```

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<tns:getPendingUserInteractions xmlns:tns="http://www.ca.com/itpam"
getRootUUID="false">
<tns:auth>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:rootUUID>6e8faae6-77a4-4461-8feb-9e5ca4d572d5</tns:rootUUID>
<tns:processID></tns:processID>
</tns:getPendingUserInteractions>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response

```
<getPendingUserInteractionsResponse xmlns="http://www.ca.com/itpam">
<userInteractions>
<userInteraction id="207" title="Task by B">
<description>Task by B</description>
</userInteraction>
<userInteraction id="208" title="Task by B">
<description>Task by B</description>
</userInteraction>
<userInteraction id="209" title="Task by B">
<description>Task by B</description>
</userInteraction>
<userInteraction id="210" title="Task by B">
<description>Task by B</description>
</userInteraction>
<userInteraction id="211" title="Task by B">
<description>Task by B</description>
</userInteraction>
<userInteraction id="212" title="Task by B">
<description>Task by B</description>
</userInteraction>
<userInteraction id="213" title="Task by B">
<description>Task by B</description>
</userInteraction>
<userInteraction id="214" title="Task by B">
<description>Task by B</description>
</userInteraction>
<userInteraction id="215" title="Task by B">
<description>Task by B</description>
</userInteraction>
<userInteraction id="216" title="Task by B">
<description>Task by B</description>
</userInteraction>
</userInteractions>
</getPendingUserInteractionsResponse>
```

Parent process calling an attached child process

Inline text

Provide the rootUUID of the parent process from the dataset:

```
<tns:getPendingUserInteractions xmlns:tns="http://www.ca.com/itpam"
getRootUUID="false">
<tns:auth>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:rootUUID>6e8faae6-77a4-4461-8feb-9e5ca4d572d5</tns:rootUUID>
<tns:processID></tns:processID>
</tns:getPendingUserInteractions>
```

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<tns:getPendingUserInteractions xmlns:tns="http://www.ca.com/itpam"
getRootUUID="false">
<tns:auth>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:rootUUID>6e8faae6-77a4-4461-8feb-9e5ca4d572d5</tns:rootUUID>
<tns:processID></tns:processID>
</tns:getPendingUserInteractions>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response

```
<getPendingUserInteractionsResponse xmlns="http://www.ca.com/itpam">
<userInteractions>
<userInteraction id="176" title="Task by B">
<description>Task by B</description>
</userInteraction>
<userInteraction id="177" title="Task by B">
<description>Task by B</description>
</userInteraction>
<userInteraction id="178" title="Task by B">
<description>Task by B</description>
</userInteraction>
<userInteraction id="179" title="Task by B">
<description>Task by B</description>
</userInteraction>
<userInteraction id="180" title="Task by B">
<description>Task by B</description>
</userInteraction>
<userInteraction id="181" title="Task by B">
<description>Task by B</description>
</userInteraction>
<userInteraction id="182" title="Task by B">
<description>Task by B</description>
</userInteraction>
<userInteraction id="183" title="Task by B">
<description>Task by B</description>
</userInteraction>
<userInteraction id="184" title="Task by B">
<description>Task by B</description>
</userInteraction>
<userInteraction id="185" title="Task by B">
<description>Task by B</description>
</userInteraction>
</userInteractions>
</getPendingUserInteractionsResponse>
```

<*userInteractions*> specifies the ID and title of every pending user interaction.

getProcessLogs

Use this method to get the logs of a process instance.

Message Levels

CA Process Automation process instance logs display the following message levels:

- Error =4
- Warning =3
- Notice =2
- Normal =1

Message Categories

CA Process Automation process instances have the following message categories:

Process

Logs messages for a process instance.

Operator

Logs messages for operators in process.

Handler

Logs messages when lane change or exception handler is invoked.

Response

Logs messages when agents send a response back to the Orchestrator.

Custom

When you log messages without any category, these messages are grouped in the Custom category. You can provide the category as Custom to retrieve such messages.

Custom Messages

You can define your own categories for logging. For example:

```
logEvent(3, "SD_MESSAGES", "this messages is logged for Service Desk operators
"
```

To retrieve such messages, specify the category as SD_MESSAGES in the request.

To retrieve log messages

Use the following SOAP request:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<tns:getProcessLogs xmlns:tns="http://www.ca.com/itpam">
<tns:auth>
<!--xsd:Choice Type-->
<tns:token>token__</tns:token>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:processID>72</tns:processID>
<tns:filter>
<tns:level>2</tns:level>
<tns:categories>
<tns:category>process</tns:category>
<tns:category>operator</tns:category>
<tns:category>handler</tns:category>
<tns:category>response</tns:category>
<tns:category> SD_MESSAGES </tns:category>
</tns:categories>
</tns:filter>
</tns:getProcessLogs>
</SOAP-ENV:Body></SOAP-ENV:Envelope>
```

<level> </level> can have a value of error, warning, notice, and normal.

If the user specifies the level as “normal”, all messages with a level of normal and above are retrieved.

When the error is at the top-most level and the user specifies the level as “error”, only messages with the “error” level are retrieved.

If the user specifies an invalid level, it is treated as normal (the lowest level).

To retrieve logs with multiple categories:

```
<tns:categories>
  <tns:category>userMsg</tns:category>
  <tns:category>process</tns:category>
</tns:categories>
```

In this example, all messages with the category “userMsg” and “process” are retrieved. Only log messages that match the level and one of the categories are returned.

Example

```
<tns:getProcessLogs xmlns:tns="http://www.ca.com/itpam">
<tns:auth>
<!--xsd:Choice Type-->
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:processID>72</tns:processID>
<tns:filter>
<tns:level>2</tns:level>
<tns:categories>
<tns:category>process</tns:category>
<tns:category>operator</tns:category>
<tns:category>handler</tns:category>
<tns:category>response</tns:category>
</tns:categories>
</tns:filter>
</tns:getProcessLogs>
```

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<tns:getProcessLogs xmlns:tns="http://www.ca.com/itpam">
<tns:auth>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:processID>72</tns:processID>
<tns:filter>
<tns:level>2</tns:level>
<tns:categories>
<tns:category>process</tns:category>
<tns:category>operator</tns:category>
<tns:category>handler</tns:category>
<tns:category>response</tns:category>
</tns:categories>
</tns:filter>
</tns:getProcessLogs>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Response


```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<getProcessLogsResponse xmlns="http://www.ca.com/itpam">
<processLogs>
<processlog category="Process" level="Notice">
<logMessage>'LaneChangeHandler_30_72' instance was created.</logMessage>
<time>2012-03-01T13:18:43.248+05:30</time>
</processlog>
<processlog category="Process" level="Notice">
<logMessage>'LaneChangeHandler_30_72' is in 'Queued' state.</logMessage>
<time>2012-03-01T13:18:43.248+05:30</time>
</processlog>
<processlog category="Process" level="Notice">
<logMessage>Process started at '03/01/2012 13:18:43' by 'pamadmin'.</logMessage>
<time>2012-03-01T13:18:43.592+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Start_Script_1' is enabled following 'Start_1'.</logMessage>
<time>2012-03-01T13:18:43.639+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Start_1' is 'Completed' on 'Current Server'.</logMessage>
<time>2012-03-01T13:18:43.639+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>A service request was sent for 'Start_Script_1'.</logMessage>
<time>2012-03-01T13:18:44.123+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Start_Script_1' is 'Running' on 'Current Server'.</logMessage>
<time>2012-03-01T13:18:44.139+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Start_Script_1' is 'Completed'</logMessage>
<time>2012-03-01T13:18:47.904+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Start_Script_1_1' is enabled following 'Start_Script_1'.</logMessage>
<time>2012-03-01T13:18:47.935+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Lane_Change_1' was reset.</logMessage>
<time>2012-03-01T13:18:47.982+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Calculation_1' was reset.</logMessage>
<time>2012-03-01T13:18:47.998+05:30</time>
```

```
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Lane_Change_2' was reset.</logMessage>
<time>2012-03-01T13:18:48.014+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Calculation_1_1' was reset.</logMessage>
<time>2012-03-01T13:18:48.029+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Lane_Change_3' was reset.</logMessage>
<time>2012-03-01T13:18:48.029+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Calculation_1_2' was reset.</logMessage>
<time>2012-03-01T13:18:48.045+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Calculation_1' is enabled following 'Lane_Change_1'.</logMessage>
<time>2012-03-01T13:18:48.045+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Lane_Change_1' is 'Completed' on 'Current Server'.</logMessage>
<time>2012-03-01T13:18:48.060+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>A service request was sent for 'Calculation_1'.</logMessage>
<time>2012-03-01T13:18:48.092+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Calculation_1' is 'Running' on 'Current Server'.</logMessage>
<time>2012-03-01T13:18:48.092+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Calculation_1' is 'Completed'</logMessage>
<time>2012-03-01T13:18:48.576+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>A service request was sent for 'Start_Script_1_1'.</logMessage>
<time>2012-03-01T13:18:48.639+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Start_Script_1_1' is 'Running' on 'Current Server'.</logMessage>
<time>2012-03-01T13:18:48.654+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Start_Script_1_1' is 'Completed'</logMessage>
<time>2012-03-01T13:18:49.060+05:30</time>
</processlog>
```

```
<processlog category="Operator" level="Notice">
<logMessage>'Start_Script_1_2' is enabled following
'Start_Script_1_1'.</logMessage>
<time>2012-03-01T13:18:49.076+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Lane_Change_1' was reset.</logMessage>
<time>2012-03-01T13:18:49.154+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Calculation_1' was reset.</logMessage>
<time>2012-03-01T13:18:49.170+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Lane_Change_2' was reset.</logMessage>
<time>2012-03-01T13:18:49.201+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Calculation_1_1' was reset.</logMessage>
<time>2012-03-01T13:18:49.217+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Lane_Change_3' was reset.</logMessage>
<time>2012-03-01T13:18:49.217+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Calculation_1_2' was reset.</logMessage>
<time>2012-03-01T13:18:49.248+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Calculation_1_1' is enabled following 'Lane_Change_2'.</logMessage>
<time>2012-03-01T13:18:49.248+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Lane_Change_2' is 'Completed' on 'Current Server'.</logMessage>
<time>2012-03-01T13:18:49.248+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>A service request was sent for 'Calculation_1_1'.</logMessage>
<time>2012-03-01T13:18:49.279+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Calculation_1_1' is 'Running' on 'Current Server'.</logMessage>
<time>2012-03-01T13:18:49.295+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Calculation_1_1' is 'Completed'</logMessage>
<time>2012-03-01T13:18:49.639+05:30</time>
</processlog>
```

```
<processlog category="Operator" level="Notice">
<logMessage>A service request was sent for 'Start_Script_1_2'.</logMessage>
<time>2012-03-01T13:18:49.701+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Start_Script_1_2' is 'Running' on 'Current Server'.</logMessage>
<time>2012-03-01T13:18:49.701+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Start_Script_1_2' is 'Completed'</logMessage>
<time>2012-03-01T13:18:50.123+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Start_Script_1_3' is enabled following
'Start_Script_1_2'.</logMessage>
<time>2012-03-01T13:18:50.154+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Lane_Change_1' was reset.</logMessage>
<time>2012-03-01T13:18:50.217+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Calculation_1' was reset.</logMessage>
<time>2012-03-01T13:18:50.248+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Lane_Change_2' was reset.</logMessage>
<time>2012-03-01T13:18:50.264+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Calculation_1_1' was reset.</logMessage>
<time>2012-03-01T13:18:50.279+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Lane_Change_3' was reset.</logMessage>
<time>2012-03-01T13:18:50.310+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Calculation_1_2' was reset.</logMessage>
<time>2012-03-01T13:18:50.326+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Calculation_1_2' is enabled following 'Lane_Change_3'.</logMessage>
<time>2012-03-01T13:18:50.342+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Lane_Change_3' is 'Completed' on 'Current Server'.</logMessage>
<time>2012-03-01T13:18:50.342+05:30</time>
</processlog>
```

```
<processlog category="Operator" level="Notice">
<logMessage>A service request was sent for 'Calculation_1_2'.</logMessage>
<time>2012-03-01T13:18:50.373+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Calculation_1_2' is 'Running' on 'Current Server'.</logMessage>
<time>2012-03-01T13:18:50.373+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Calculation_1_2' is 'Completed'</logMessage>
<time>2012-03-01T13:18:50.842+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>A service request was sent for 'Start_Script_1_3'.</logMessage>
<time>2012-03-01T13:18:51.107+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Start_Script_1_3' is 'Running' on 'Current Server'.</logMessage>
<time>2012-03-01T13:18:51.123+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Start_Script_1_3' is 'Completed'</logMessage>
<time>2012-03-01T13:18:55.982+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Normal_Stop_1' is enabled following 'Start_Script_1_3'.</logMessage>
<time>2012-03-01T13:18:56.014+05:30</time>
</processlog>
<processlog category="Operator" level="Notice">
<logMessage>'Normal_Stop_1' is 'Completed' on 'Current Server'.</logMessage>
<time>2012-03-01T13:18:56.060+05:30</time>
</processlog>
<processlog category="Process" level="Notice">
<logMessage>Process is 'Completed'.</logMessage>
<time>2012-03-01T13:18:56.060+05:30</time>
</processlog>
</processLogs>
</getProcessLogsResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

getProcessStatus

Use this method to view the state of a process instance.

Example

```
<tns:getProcessStatus xmlns:tns="http://www.ca.com/itpam">
<tns:flow>
<!--xsd:Choice Type-->
<tns:ROID>489</tns:ROID>
<tns:action>check</tns:action>
<tns:auth>
<!--xsd:Choice Type-->
<tns:token>token__</tns:token>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
</tns:flow>
</tns:getProcessStatus>
```

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<tns:getProcessStatus xmlns:tns="http://www.ca.com/itpam">
<tns:flow>
<tns:ROID>489</tns:ROID>
<tns:tagId>z34dsf5c7-xagb-4g5d-74g5-bd5c4we2378f</tns:tagId>
<tns:action>check</tns:action>
<tns:auth>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
</tns:flow>
</tns:getProcessStatus>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

You can include one of the following:

- <*ROID*> specifies the ROID of the process instance that you are viewing the state of.
- <*tagId*>: The user passes the tagId of the start request form to view.

<*action*> specifies the action to perform on the specified instance; the value of this tag must be "check".

SOAP Responses

Completed Process

```
<CheckFlowResponse xmlns="http://www.ca.com/itpam">
<CheckFlowResult>
<ROID>489</ROID>
<flow-state>Completed</flow-state>
<submission-time>2012-02-01T20:15:41.515+05:30</submission-time>
<sender-address>none</sender-address>
<connection-id>none</connection-id>
<message>Flow state received</message>
<params>
<param name="RuntimeROID">489</param>
<param name="Result">1</param>
<param name="ServerId">b9c4b8de-b74b-44b5-b2d2-1deb6e3dc872</param>
<param name="effectiveUser">pamadmin</param>
<param name="InstanceUUID">f120811e-3895-41a2-abbb-3458c19a51c8</param>
<param name="UserName">pamadmin</param>
<param name="ProcessName">/SOAPTest/P1</param>
<param name="ServerName">lodivsa204.ca.com</param>
<param name="DisplayName">P1</param>
<param name="TouchpointName">Orchestrator</param>
<param name="CallerUser">pamadmin</param>
<param name="EndDate">2012-02-01 16:30:24</param>
<param name="StartDate">2012-02-01 16:30:24</param>
<param name="ScheduledStartTime">2012-02-01 00:00:00</param>
<param name="rootUUID">f120811e-3895-41a2-abbb-3458c19a51c8</param>
<param name="EndTime">2012-02-01 16:30:24</param>
<param name="StartTime">2012-02-01 16:30:24</param>
<param name="ProcessAction">start</param>
<param name="InstanceName">P1_489</param>
</params>
</CheckFlowResult>
</CheckFlowResponse>
```

<*flow-state*> returns the state of the process instance.

Nonexistent or Archived Process

The operator fails with a fault and returns this response:

```
<Fault xmlns="http://schemas.xmlsoap.org/soap/envelope/">
<faultcode>SOAP-ENV:Server</faultcode>
<faultstring>The instance does not exist. It may have been archived. Refresh the
screen.</faultstring>
</Fault>
```

Invalid Action

The operator fails with this fault:

```
<Fault xmlns="http://schemas.xmlsoap.org/soap/envelope/">
<faultcode>SOAP-ENV:Server</faultcode>
<faultstring>Invalid flow action: c</faultstring>
</Fault>
```

Invalid ROID

The operator fails with this fault:

```
<Fault xmlns="http://schemas.xmlsoap.org/soap/envelope/">
<faultcode>SOAP-ENV:Server</faultcode>
<faultstring>error retrieving flow state</faultstring>
</Fault>
```

getStartRequestForm

Use this method to retrieve information about a start request form that is based on the name and location of the start request form from the CA Process Automation library.

Example

```
<tns:getStartRequestForm xmlns:tns="http://www.ca.com/itpam"
getParamSequence="true">
<tns:auth>
<!--xsd:Choice Type-->
<tns:token>token__</tns:token>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:objLocation>
<tns:name>MYSRF</tns:name>
<tns:path>/myfolder/</tns:path>
</tns:objLocation>
</tns:getStartRequestForm>
```

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Header/><SOAP-ENV:Body>
<tns:getStartRequestForm xmlns:tns="http://www.ca.com/itpam"
getParamSequence="true">
<tns:auth>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:objLocation>
<tns:name>MYSRF</tns:name>
<tns:path>/TENSU03/</tns:path>
</tns:objLocation>
</tns:getStartRequestForm>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

<*name*> is the name of the start request form.

<*path*> is the absolute path of the folder where the start request form resides.

SOAP Response

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<getStartRequestFormResponse xmlns="http://www.ca.com/itpam">
<refProcess>/Joe/Process</refProcess>
<pages hasComplexType="false">
<page name="Page">
<itpamString isReadOnly="false" maxlength="2147483647" minlength="0" name="Var_0"
sequenceNo="0">
<label>Text Field</label>
<description/>
<value/>
</itpamString>
<itpamString isReadOnly="false" maxlength="2147483647" minlength="0" name="Var_1"
sequenceNo="1">
<label>Text Field</label>
<description/>
<value/>
</itpamString>
</page>
<page name="System"/>
</pages>
</getStartRequestFormResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

<*refProcess*> shows the name and path of the process that is attached with this start request form.

getStartRequestForms

Use this method to search the start request forms in a CA Process Automation library. A search is based on the path and keyword provided.

Example

With /root as the lookup path

```
<tns:getStartRequestForms xmlns:tns="http://www.ca.com/itpam">
<tns:auth>
<!--xsd:Choice Type-->
<tns:token>token__</tns:token>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:filter>
<tns:lookUpPath isRecursive="true">/</tns:lookUpPath>
<tns:keywords>
<tns:keyword>ABC</tns:keyword>
</tns:keywords>
</tns:filter>
</tns:getStartRequestForms>
```

To obtain all the start request forms in the environment

Provide / and remove the Keyword__ from the Keyword tags:

```
<tns:getStartRequestFormsResponse xmlns:tns="http://www.ca.com/itpam">
<startRequests>
<tns:startRequest name="Start Purchase Request Form" refPath="/PAM Hardware
Procurement/">
<tns:description/>
</tns:startRequest>
<tns:startRequest name="Asset_Install_SW_SRF" refPath="/sinra1/CA ITAM/SRF/">
<tns:description>Database Object</tns:description>
</tns:startRequest>
<tns:startRequest name="Asset_MAC_HW_SRF" refPath="/sinra1/CA ITAM/SRF/">
<tns:description>Database Object</tns:description>
</tns:startRequest>
<tns:startRequest name="Asset_Transfer_SRF" refPath="/sinra1/CA ITAM/SRF/">
<tns:description>Database Object</tns:description>
</tns:startRequest>
<tns:startRequest name="Asset_Disposal_HW_SRF" refPath="/sinra1/CA ITAM/SRF/">
<tns:description>Database Object</tns:description>
</tns:startRequest>
<tns:startRequest name="Asset_Install_HW_Transferred_SRF" refPath="/sinra1/CA
ITAM/SRF/">
<tns:description>Database Object</tns:description>
</tns:startRequest>
<tns:startRequest name="Asset_Install_HW_SRF" refPath="/sinra1/CA ITAM/SRF/">
<tns:description>Database Object</tns:description>
</startRequest>
<tns:startRequest name="~Start Request Form" refPath="/Package Demo/">
<tns:description/>
</tns:startRequest>
<tns:startRequest name="~Start Request Form" refPath="/Jack/">
<tns:description/>
</tns:startRequest>
</tns:startRequests>
</tns:getStartRequestFormsResponse>
```

Perform a keyword search

You can search start request forms with keywords that are associated with them. For example, you can perform a search to find if *SRF1* and *SRF2* are associated with the keyword "ABC" in the folder */myfolder*.

```
<tns:getStartRequestForms xmlns:tns="http://www.ca.com/itpam">
<tns:auth>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:filter>
<tns:lookUpPath isRecursive="true">/myfolder</tns:lookUpPath>
<tns:keywords>
<tns:keyword>ABC</tns:keyword>
</tns:keywords>
</tns:filter>
</tns:getStartRequestForms>
```

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<tns:getStartRequestForms xmlns:tns="http://www.ca.com/itpam">
<tns:auth>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:filter>
<tns:lookUpPath isRecursive="true">/</tns:lookUpPath>
<tns:keywords>
<tns:keyword>ABC</tns:keyword>
</tns:keywords>
</tns:filter>
</tns:getStartRequestForms>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

This tag is present in the <filter> tag:

- <*lookUpPath*> provides the path under which the user wants to search the start request forms. One attribute in this tag is recursive. If a user wants to search recursively under the provided path, then set this attribute true. If a user wants to search only under the provided path, set this attribute as false. Provide the complete path (as /Folder/Folder1) as the value of this tag in which user wants to search.

SOAP Response

```
<getStartRequestFormsResponse xmlns="http://www.ca.com/itpam">
<startRequests>
<startRequest name="SRF1" refPath="/TENSU03/">
<description/>
</startRequest>
<startRequest name="SRF2" refPath="/TENSU03/">
<description/>
</startRequest>
</startRequests>
</getStartRequestFormsResponse>
```

For every start request form that fulfills the criteria for the search, there is a <startRequest> under the <*startRequests*> tag. This tag contains the name, reference path, and description of the start request form in the SOAP response.

ImportObject

Use this method to import objects into a CA Process Automation Library.

Note: Ensure that the maximum XML size that you want to import is 250 MB.

Example

```
<tns:ImportObject xmlns:tns="http://www.ca.com/itpam">
<tns:importFileParam>
<tns:auth>
<!--xsd:Choice Type-->
<tns:token>token__</tns:token>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:importLocation>importLocation__</tns:importLocation>
<tns:localSourceLocation>localSourceLocation__</tns:localSourceLocation>
<tns:isSetCurrent>isSetCurrent__</tns:isSetCurrent>
<tns:isMakeAvailable>isMakeAvailable__</tns:isMakeAvailable>
<tns:overwriteAction>overwriteAction__</tns:overwriteAction>
</tns:importFileParam>
<!--This SOAP request can have MIME contents-->
</tns:ImportObject>
```

<*isSetCurrent*> specifies whether to mark the imported objects as the current versions (optional).

<*isMakeAvailable*> specifies whether to mark imported custom operators as available (optional).

After you provide values in the inline text, create a MIME attachment for the file you specify in the <*localsourcelocation*> tag in the SOAP request. To add a MIME attachment, in the Properties panel under the MIME attachments, click Add, then provide appropriate values for:

Content Type

Type of content the MIME attachment is carrying. For example, text/xml.

Content ID

ID that the attachment is uniquely identified with. For example, 11111.

File URL

Path of the attachment. This path can be the path of the .xml to be imported. For example, C:\\SOAPExportedObjects.xm.

Save and run the process.

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<tns:ImportObject xmlns:tns="http://www.ca.com/itpam">
<tns:importFileParam>
<tns:auth>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:importLocation>/</tns:importLocation>
<tns:localSourceLocation>"C:\\SOAPExportedObjects.xml"</tns:localSourceLocation>
<tns:isSetCurrent>true</tns:isSetCurrent>
<tns:isMakeAvailable>true</tns:isMakeAvailable>
<tns:overwriteAction>incrementObjectVersion</tns:overwriteAction>
</tns:importFileParam>
<!--This request can have MIME contents-->
</tns:ImportObject>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

<*importLocation*> specifies the location in the CA Process Automation library where you want to import the objects.

<*isSetCurrent*> specifies whether to mark the imported objects as the current versions (optional). The default value is false.

<*isMakeAvailable*> specifies whether to mark imported custom operators as available (optional). The default value is false.

<overwriteAction>

If an object with the same name already exists in the Library, you can select one of the following options for <*overwriteAction*>:

- incrementObjectVersion - Import as a new version and keep the existing object.
- replaceObject - Import and replace the existing object.
- skipImport - Do not import objects with the same name as an existing object.

For example:

```
<tns:ImportObject xmlns:tns="http://www.ca.com/itpam">
<tns:importFileParam>
<tns:auth>
<tns:user>Joe</tns:user>
<tns:password>thisismypassword</tns:password>
<tns:auth>
<tns:importLocation>/ABC</tns:importLocation>
<tns:localSourceLocation>"C://abc.xml"</tns:localSourceLocation>
<tns:isSetCurrent>true</tns:isSetCurrent>
<tns:isMakeAvailable>true</tns:isMakeAvailable>
<tns:overwriteAction>skipImport</tns:overwriteAction>
</tns:importFileParam>
<!--This SOAP request can have MIME contents-->
</tns:ImportObject>
```

Result:

```
<importFileResponse xmlns="http://www.ca.com/itpam">
<messages>
<message>Could not import object CustomIcon. Another object with the same name
as "CustomIcon" already exists.</message>
<message>Could not import object P1. Another object with the same name as "P1"
already exists.</message>
<message>Could not import object Process. Another object with the same name
as "Process" already exists.</message>
<message>Could not import object Agenda. Another object with the same name
as "Agenda" already exists.</message>
<message>Could not import object Resources. Another object with the same name
as "Resources" already exists.</message>
<message>Could not import object SOAPMethods. Another object with the same
name as "SOAPMethods" already exists.</message>
<message>Could not import object ProcessWatch. Another object with the same
name as "ProcessWatch" already exists.</message>
<message>Could not import object Calendar. Another object with the same name
as "Calendar" already exists.</message>
<message>Could not import object SOAPModule_NegativeCases. Another object
with the same name as "SOAPModule_NegativeCases" already exists.</message>
```

```
<message>Could not import object IRF. Another object with the same name as
"IRF" already exists.</message>
<message>Could not import object SOAPOperators_DynamicParams. Another object
with the same name as "SOAPOperators_DynamicParams" already
exists.</message>
<message>Could not import object Interaction Request Form. Another object
with the same name as "Interaction Request Form" already exists.</message>
<message>Could not import object SRF. Another object with the same name as
"SRF" already exists.</message>
<message>Could not import object SOAPTest_Trigger. Another object with the
same name as "SOAPTest_Trigger" already exists.</message>
<message>Could not import object CustomOperator. Another object with the same
name as "CustomOperator" already exists.</message>
<message>Could not import object Process_2. Another object with the same name
as "Process_2" already exists.</message>
<message>Could not import object Dataset. Another object with the same name
as "Dataset" already exists.</message>
<message>Could not import object Start Request Form. Another object with the
same name as "Start Request Form" already exists.</message>
<message>Could not import object GlobalVars. Another object with the same name
as "GlobalVars" already exists.</message>
<message>Could not import object Package. Another object with the same name
as "Package" already exists.</message>
</messages>
</importFileResponse>
```

SOAP Response

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<importFileResponse xmlns="http://www.ca.com/itpam">
<successMessage>The specified object imported successfully.</successMessage>
</importFileResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

<*successMessage*> displays a successful import message.

Common Tags for Web Services Methods

The following common tags apply to numerous Web services methods:

The <pages> Tag (see page 166)

This tag is used in the following Web services:

- getPendingInteractionRequestForm Method (see page 137)
- getStartRequestForm Method (see page 156)

The <attachments> Tag (see page 169)

This tag is used in the following Web services:

- deleteAttachments Method (see page 115)
- executeStartRequest Method (see page 122)
- getAttachments Method (see page 131)
- excuteProcess Method (see page 119)

The <params> Tag (see page 170)

This tag is used in the following Web services:

- checkStartRequestStatus Method (see page 107)
- executePendingInformation Method (see page 116)
- executeProcess Method (see page 119)
- executeStartRequest Method (see page 122)
- getProcessStatus Method (see page 153)

The <options> Tag (see page 170)

This tag is used in the following Web services:

- executeProcess Method (see page 119)
- executeStartRequest Method (see page 122)

The <page> and <pages> Tags

In a SOAP response, the <pages> tag represents the pages in a start request form/interaction request form. For every page in the start request form/interaction request form, there is a <page> tag under the <pages> tag.

The parameters in that page are shown under the <page> tag. The <page> tag contains an attribute name which shows the name of the page.

A corresponding tag is under <Page> for most of the types of variables in the form dataset.

Mapping between dataset variables and tags under <Page>.

- Boolean\itpamBoolean
- Date\itpamDate
- Double\itpamDouble
- Integer\itpamInt
- Long\itpamLong
- Password\itpamPassword
- String\itpamString

Note: CA Technologies does not support ValueMaps, Object References or Arrays through Web services.

The attributes of the parameter element include:

- Validation attributes of the parameter
- Name of the parameter

The following appear under this node:

- Label
- Value (not returned in case of password)
- Description (returned if not blank)
- Default value (returned if not blank)

The <option> tag displays with the options you specify (if you specify predefined values for a parameter in a start request form/interaction request form). The isallowOtherValue attribute of <option> tag is true if the user can specify a value other than what is provided under option.

Example

```
<pages hasComplexType="false">
            <page name="Parameters">
               <itpamString isReadOnly="false" maxlength="254" minlength="0"
name="Var_0" sequenceNo="0">
                  <label>Var_0</label>
                  <description/>
                  <value>pink</value>
<defaultValue>op10</defaultValue>
                  <options  isallowOtherValue="false">
                     <op name="option1">op1</op>
                     <op name=" option2">op2</op>
                  </options>
               </itpamString>
               <itpamInt isReadOnly="false" maxval="2147483647"
minval="-2147483648" name="Var_4" sequenceNo="4">
                  <label>Var_4</label>
                  <description/>
                  <value>0</value>
                  <defaultValue>0</defaultValue>
               </itpamInt>
 <itpamLong isReadOnly="false" maxval="9223372036854775807"
minval="-9223372036854775808" name="Var_1" sequenceNo="1">
                  <label>Var_1</label>
                  <description/>
                  <value>0</value>
                  <defaultValue>0</defaultValue>
               </itpamLong>
               <itpamDouble isReadOnly="false" maxval="1.7976931348623157E308"
minval="-1.7976931348623157E308" name="Var_2" sequenceNo="2">
```

```
            <label>Var_2</label>
            <description/>
            <value>0.0</value>
            <defaultValue>0.0</defaultValue>
         </itpamDouble>
         <itpamDate isReadOnly="false" name="Var_3" sequenceNo="3">
            <label>Var_3</label>
            <description/>
         </itpamDate>
         <itpamBoolean isReadOnly="false" name="Var_5" sequenceNo="5">
            <label>Var_5</label>
            <description/>
            <value>false</value>
            <defaultValue>false</defaultValue>
         </itpamBoolean>
      </page>
      <page name="User Prompt"/>
      <page name="System"/>
   </pages>            <page name="System"/>
   </pages>
```

The <attachments> Tag

The <attachments> tag provides meta information about attachment. CA Process Automation supports adding multiple attachments to the executeProcess and executeStartRequestForm methods.

Example

```
<attachments attachmentsParamName="?">
    <attachment
         <attachmentID> </attachmentID>
             <name>?</name>
    <attachment>
```

<*attachmentID*> provides the content Id of attachment.

<*name*> specifies the name of attachment.

<*attachmentsParamName = "paramName"*> specifies the name of variable which appears in the process dataset of the executed process instance. The attachment can be accessed using the *paramName* name variable.

Important! When integrating into CA Process Automation using Web services, the Web service stubs generated using common third-party tools (such as wsdl2java) do not cleanly generate code for handling attachments. To integrate into CA Process Automation using Web services that manipulate attachments, see the "CA IT PAM Web Utilities" section of the "CA IT PAM Best practices" page. This page is linked from the CA IT PAM home page on support.ca.com.

The <params> Tag

To provide a parameter, create one <params> tag. You can provide a list of parameters under the <params> tag in the request.

Example

```
<params>
     <param name="vendorName">CA </param>
     <param name="quantity"> 4</param>
     <param name="start Date">10/30/2000</param>
     <param name="approvalRequired">false</param>
</params>
```

The name attribute specifies the name of the parameter.

Value specifies the value of the parameter in the value attribute.

- If the parameter exists in the dataset of the CA Process Automation object, the value of the parameter is adjusted with new value.
- If the parameter does not exist, a new variable is created with the name and value provided. The type of the variable depends upon the value provided.

The <options> Tag

The <options> tag contains the scheduling parameters.

Example

```
<options>
    <startDate> 2002-09-24</startDate>
    <startTime> 09:00:00 </startTime>
 </options>
```

The process executes according to the start date and start time provided.

Appendix A: HTTP Status and Error Codes

Every HTTP response contains an HTTP status code. Successful HTTP response code numbers range from 200 to 399. Standard HTTP error response code numbers range from 400 to 599.

The following list shows the successful and error code numbers that the REST API returns for each of the HTTP methods.

GET (single)

200, 400, 401, 404, 409

GET (collection)

200, 400, 401

PUT

200, 400, 401, 404, 409

DELETE

204, 400, 401, 409

POST

201, 400, 401, 409

Note: For the PUT and DELETE operations, the API does *not* try to determine the validity of the ID initially. Instead, the API tries to run the update and delete queries directly. If an error occurs, the API returns the *409 Conflict* code. If the API returns a *404 Not Found* code in this situation, performance degrades.

In addition, the following error messages return under the following cases:

- If the HTTP request contains an invalid or inaccessible URI address, the server responds with a *404 Not Found* response code.
- If the HTTP request contains an unsupported HTTP method for a valid URI, the server responds with a *405 Method Not Allowed* response code.
- If the HTTP request requests an unsupported media type (Accept header), the server responds with a *406 Not Acceptable* response code.
- If the HTTP request sends an unsupported media type (Content-Type header), the server responds with a *415 Unsupported Media Type* response code.
- Various syntax or internal Web Server errors can return a 500 internal error.

HTTP Status and Error Codes

Every HTTP response contains an HTTP status code. Successful HTTP response code numbers range from 200 to 399. Standard HTTP error response code numbers range from 400 to 599.

The following list shows the successful and error code numbers that the REST API returns for each of the HTTP methods.

GET (single)

200, 400, 401, 404, 409

GET (collection)

200, 400, 401

PUT

200, 400, 401, 404, 409

DELETE

204, 400, 401, 409

POST

201, 400, 401, 409

Note: For the PUT and DELETE operations, the API does *not* try to determine the validity of the ID initially. Instead, the API tries to run the update and delete queries directly. If an error occurs, the API returns the *409 Conflict* code. If the API returns a *404 Not Found* code in this situation, performance degrades.

In addition, the following error messages return under the following cases:

- If the HTTP request contains an invalid or inaccessible URI address, the server responds with a *404 Not Found* response code.
- If the HTTP request contains an unsupported HTTP method for a valid URI, the server responds with a *405 Method Not Allowed* response code.
- If the HTTP request requests an unsupported media type (Accept header), the server responds with a *406 Not Acceptable* response code.
- If the HTTP request sends an unsupported media type (Content-Type header), the server responds with a *415 Unsupported Media Type* response code.
- Various syntax or internal Web Server errors can return a 500 internal error.

Known Status Codes

The following list describes the known status codes that the API returns. Other codes may exist from the web server or the CXF framework, but it depends on the type of error.

200

OK

Indicates a successful return.

201

Created

Indicates a new record.

204

No Content

Indicates an empty response body.

304

Not Modified

Indicates that the record did not update.

400

Bad Request

Indicates that an error occurred due to a user or backend server issue.

401

Unauthorized

Indicates a function Access error or any authentication failure.

404

Not Found

Indicates that a record is not found.

405

Method Not Allowed

Indicates an unsupported HTTP method.

406

Not Acceptable

Indicates an unsupported requested format.

409

Conflict

Indicates that multiple records were found for the given identifier.

415

Unsupported Media Type

Indicates that the provided format is not supported.

500

Internal Server Error

Indicates an error on the server or CXF framework.

Index