CA Release Automation

Reference

5.0

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: Reference

CA Release Automation APIs enable users to retrieve data and execute processes and releases designed and published in CA Release Automation without invoking the user interface.

CA Release Automation provides a command line interface, a web services API, and a REST API to manage process and release executions.

Command-Line Reference (see page 11)

Provides the the command-lines to invoke processes and deployments.

Open API Reference (see page 33)

Use the Open API to configure, execute, and monitor process runs.

Rest API Reference (see page 97)

Provides the APIs and DTOs to retrieve data and execute processes and releases published in CA Release Automation.

Chapter 2: Command-Line Reference

The Command-Line Interface enhanced scripts support Automation Studio processes and enables the user to perform operations in Release Operations Center without invoking either User Interface.

In Automation Studio, the nolio.cmd/sh script supports the following operations:

- Run a process.
- Stop a job run (executing process).
- Get the status of a job run.
- Create an export file of application data.

In Release Operations Center, the nolio.cmd/sh script supports the following operations:

- Create a release out of a template that is defined in the Release Operations Center.
- Update an existing release.
- Run a specific release.
- Get status of a release.
- Schedule a release.

Install the Command-Line

To enable the use of commands that perform various Automation Studio and Release Operations Center tasks without invoking the UI, install the CA Release Automation CLI.

Follow these steps:

1. Launch the installation wizard, or to get prompts for the configuration and use through the CLI, activate one of the following platform-specific executables.

   ```
   nolio_cli_linux_5_0_0_b<#>.sh
   nolio_cli_windows_5_0_0_b<#>.exe
   ```

2. Transfer the installation file to the target computer.
3. Grant “a+x” permission to the installation file:
   ```
   chmod a+x nolio_cli_linux_5_0_0_b<#>.sh
   ```

4. Execute the installation file:

   `./nolio_cli_linux_5_0_0_b<#>.sh`

   `./nolio_cli_windows_5_0_0_b<#>.exe`

5. Follow the instructions on the screen.

   The system command prompt appears.

CLI Commands

The CLI enhanced scripts support Automation Studio processes and enable the user to perform operations in Release Operations Center without invoking either User Interface.

The CLI commands are executed on one line, with a space separating each argument.

The following commands are the CLI syntax:

Windows

```
nolio.cmd [create-release | update-release | schedule-release | run-release |
get-release-status] [options...]
```

```
nolio.cmd [run-process | stop-job | get-job-status | export-application] [options...]
```

Linux

```
nolio.sh [create-release | update-release | schedule-release |run-release |
get-release-status] [options...]
```

```
nolio.sh [run-process | stop-job | get-job-status | export-application] [options...]
```

The CLI qualifications are:

- Command arguments can include spaces ONLY if the entire argument value is contained in embedded quotation marks.
- User inputs for parameters have to be referred to in the CLI, except parameters that are set for User Input but have a default value.
- Include the server defined for the process environment in the execution, when a process parameter is used.

The following format requirements relate to user input parameter values:

User input parameter values with embedded commas (',') are supported only if a backslash ('\') precedes the comma. If a backslash and comma ('\,') are embedded in the parameter value, two backslashes ('\\') must precede the comma. See the following examples:

```
param1, a,b becomes param1, a\,b
```

```
param1, a\,b becomes param1, a\\,b
```

Embedded hyphens ('-') are supported in parameter values only if a backslash precedes the hyphen ('\'). For example:

```
param1, a-b' becomes 'param1, a\b
```

When setting the parameter values for an array:

The array values for a parameter are enclosed by a pair of square brackets ('[]').

Multiple array values must be delimited by a backslash and comma pair ('\,'), as in the following example:

```
arrayval1\,arrayval2\,arrayval3
```

Array values with embedded hyphens are allowed, while embedded commas are not.

nolio create-release

Use the create-release command to trigger the creation of a release in the Release Operations Center.

The command uses the following format:

```
nolio.cmd/sh create-release [options...]
```

-a *app*

The application name on which the operation occurs.

-e *env*

The environment name on which the operation occurs.

-h *help*

Display list of options.

-l *skip-validation*

(Optional) Skip the validation of the release if all its steps are assigned to the requested environment.

-m *template*

The template name from which the release is created.

-n *async*

(Optional) Run asynchronously (does not wait for the execution to terminate).

-p p*assword*

The password that is used to connect to CA Release Automation.

-r *release*

Name of the release to be created.

-t *timeout*

(Optional) Timeout for execution (seconds).

-u *user*

The login username for the Application Release.

-y *release-type*

Release type <Minor/Major/Emergency>.

nolio update-release

To update a release in the Release Operations Center, use the update-release command.

The command uses the following format:

```
nolio.cmd/sh update-release [options...]
```

-a *app*

The application name on which the operation occurs.

-c *conf-file*

(Optional) Path to the configuration file.

-e *env*

The environment name on which the operation occurs.

-h *help*

Display list of options.

-i *release-id*

The unique release id.

-p p*assword*

The password that is used to connect to CA Release Automation.

-r *release*

Name of the release to be created.

-u *user*

The login username for the Application Release.

-v *version*

Release version.

nolio schedule-release

To schedule a release in the Release Operations Center, use the schedule-release command.

The command uses the following format:

```
nolio.cmd/sh schedule-release [options...]
```

Note: Schedule-release can be used with other commands, in the same command line. The command must not appear before create-release or update-release and must not appear after run-release or get-release-status.

-a *app*

The application name on which the operation occurs.

-d *schduled-date*

The schedule date that is formatted as dd/mm/yy and calculated using the timezone set in CA Release Automation.

-e *env*

The environment name on which the operation occurs.

-ed *estimated-duration*

(Optional) The estimated duration for the release in minutes. The default value is two hours.

-h *help*

Display list of options.

-i *release-id*

The unique release id.

-p p*assword*

The password that is used to connect to CA Release Automation.

-r *release*

Name of the release to be created.

-ti *schedule-time*

The scheduled time that is formatted as HH:mm and calculated according to the timezone set in CA Release Automation.

-u *user*

The login username for the Application Release.

-v *version*

Release version.

nolio run-release

To run a release in the Release Operations Center, use the run-release command.

The command uses the following format:

```
nolio.cmd/sh run-release [options...]
```

-a *app*

The application name on which the operation occurs.

-e *env*

The environment name on which the operation occurs.

-h *help*

Display list of options.

-i *release-id*

The unique release id.

-n *async*

(Optional) Run asynchronously (does not wait for the execution to terminate).

-p p*assword*

The password that is used to connect to CA Release Automation.

-r *release*

Name of the release to be created.

-t *timeout*

(Optional) Timeout for execution (seconds).

-u *user*

The login username for the Application Release.

-v *version*

Release version.

nolio get-the-release-status

To get the status of a release in the Release Operations center, use the get-the-release-status command.

The command uses the following format:

```
nolio.cmd/sh get-the-release-status
```

-a *app*

The application name on which the operation occurs.

-e *env*

The environment name on which the operation occurs.

-h *help*

Display list of options.

-i *release-id*

The unique release id.

-p p*assword*

The password that is used to connect to CA Release Automation.

-r *release*

Name of the release to be created.

-u *user*

The login username for the Application Release.

-v *version*

Release version.

nolio run-process

To run a process in Automation Studio, use the run-process command.

The command uses the following format:

```
nolio.cmd/sh run-process [options...]
```

-a *app*

The application name on which the operation occurs.

-b *tag*

(Optional) Name of the process tag to run. If no name is given, the latest published version is used.

-c *conf-file*

(Optional) Path to the configuration file.

-e *env*

The environment name on which the operation occurs.

-f *flow*

The process name to run.

-h *help*

Display list of options.

-j *job-name*

(Optional) Set a display for the executed job.

-n *async*

(Optional) Run asynchronously (does not wait for the execution to terminate).

-p p*assword*

The password that is used to connect to CA Release Automation.

-r *params*

(Optional) A list of parameters to set. Format to use: {param1, val1, param2, val2 …}).

-s *servers*

(Optional) A list of servers to run the process on, in the format {server1, server2, ...} If none, the process is executed on all servers in the relevant Server type.

-t *timeout*

(Optional) Timeout for execution (seconds).

-u *user*

The login username for the Application Release.

-x *keep-job*

(Optional) Keep process run active in case process run is paused due to failure (normal intervention is required).

nolio stop-job

To stop the execution of a job in Automation Studio, use the stop-job command.

The command uses the following format:

```
nolio.cmd/sh stop-job [options...]
```

-h *help*

Display list of options.

-i *job-id*

The unique job (executed process) id.

-p p*assword*

The password that is used to connect to CA Release Automation.

-u *user*

The login username for the Application Release.

nolio get-job-status

To get the status of a job in Automation Studio, use the get-job-status command.

The command uses the following format:

```
nolio.cmd/sh get-job-status [options...]
```

-h *help*

Display list of options.

-i *job-id*

The unique job (executed process) id.

-p p*assword*

The password that is used to connect to CA Release Automation.

-u *user*

The login username for the Application Release.

nolio export-application

To export an application in Automation Studio, use the export-application command.

The command uses the following format:

```
nolio.cmd/sh export-application [options..]
```

-a *app-name*

(Optional) A list of application names to be exported. Format to use {app_name1,app_name2,...} If none, all.

-f *file-path*

Path of the file.

-h *help*

Display list of options.

-p p*assword*

The password that is used to connect to CA Release Automation.

-u *user*

The login username for the Application Release.

Command-Line Examples

The following examples provide an understanding of how to set parameters using the CLI.

The examples in this document use the Windows command file.

```
>nolio.cmd run-process –u superuser –p suser –a MyApp –e NY_DC_South –f "Collect logs process" -s {dcnyapp1, dcnyapp2} –r "{dcnyapp2/Install Dir,c:/Program Files,dcnyapp2/Copy Dst,c:/Temp}"
```

View Examples of Command Parameters

Run a Process with the Application Parameter

The following example illustrates the case of a CLI command for running a process with an Application Parameter:

```
>nolio.cmd run-process -u superuser -p suser -a "Test" -e "Environment for Default Architecture"  –f "delay" -r "{Application Parameters/app_param,hello cli}"
```

Note: The command overrides a parameter set as an Environment parameter as long as User Input is selected.

Two Server Types

The following example illustrates where the CLI command refers to two server types, similar to the example of the CLI with more than one parameter.

Each parameter is referenced under its server. The servers in this instance are QA_LAB1 and QA_LAB2, which are the names of the agents:

```
>nolio.cmd run-process -u superuser -p suser -a "CLI_APP" -e "Environment for CLI arch" -f " CLI_BOOL_PROC" -r "{QA_LAB1/CLI_COMP/cli_bool,true,QA_LAB2/CLI_COMP/cli_bool,false}"
```

Note: A warning is not given if you run a multiple server type process and you select only one server type to run.

Parameter Under the Default Component Folder

The following example illustrates a CLI command where the parameter is under the default component folder:

```
>nolio.cmd run-process -u superuser -p suser -a "Test" -e "Environment for Default Architecture" –f "delay" -r "{QA_LAB2/Default Component/param,hello cli 2}"
```

Parameter Under a Server Type Folder

The following example illustrates a CLI command where the parameter is under a server type folder:

```
>nolio.cmd run-process -u superuser -p suser -a "Test" -e "Environment for Default Architecture"  –f "delay" -r "{QA_LAB1/st1p,hello cli 2}"
"QA_LAB1" is the agent name; "st1p" is the name of the string parameter that actually sits under a folder called "Server Type 1".
```

To run the same process on five agents using the set parameter under Server Type 1, enter the parameter into the CLI five times with each agent name.

In the example, "hello cli 2" is the value for the string parameter.

Parameter Under a Folder in Application Parameters

The following example illustrates a CLI command where the target parameter is under a folder in the Application Parameters:

```
>nolio.cmd run-process -u superuser -p suser -a "Test" -e "Environment for Default Architecture"  –f "delay" -r "{Application Parameters/Folder/app_param,hello cli}"
```

Multiple Parameters

The following example illustrates a CLI command containing more than one parameter: *Param1,Value1,Param2,Value2:*

```
>nolio.cmd run-process -u superuser -p suser -a "Test" -e "Environment for Default Architecture" -f "delay1" -r "{Application Parameters/intp,7,QA_LAB3/Default Component/param,Hello second param}"
```

Note: If you provide part of the values for a process with multiple user-input parameters, the process automatically pauses to wait for the omitted parameters. Resume the process by providing the remaining parameters in the User Interface.

Set a Virtual Name for a Process

When running CLI executions, the triggered process is listed as “Remote Execution": <PROCESSNAME> (<DATE>).

To provide a virtual name for an execution for easier tracking in the UI, use the -j command, as in the following example:

```
>nolio.cmd run-process -u superuser -p suser -e defaultenv –f p2 -a meir -j myname
```

In the UI Administration tab, the Online Audit Report displays the virtual process name in the Run column:

Run a Process with Suspend on Failure Option

All submitted processes proceed to one of the following stages during execution:

- RUNNING/FINISHED
- PRE_FAILED
- FAILED_PAUSED

The few situations in which the process terminates are:

- The process entered a PRE_FAILED state before the process was created.
- The process entered a FAILED_PAUSED state while running.
- The process was paused from the Release Automation UI and the CLI still shows the state as RUNNING.

If a process was submitted using the '-x' command, Release Automation puts the process in a suspended state. The users can then:

- See the FAILED_PAUSE stage in the UI
- Check the execution result string using the -g command in the CLI

Using the stage or result string information, the user can fix the problem and can continue the execution by selecting RESUME in the UI.

The following example uses the -x command:

```
>nolio.cmd run-process -u superuser -p suser -e defaultenv -f p2 -a meir –x
```

Retrieve the Process Status

Once a Process ID has been generated, use the following commands to verify the process or the error that it generated:

Return a string-based result status and extra information:

```
-i <Process ID>
```

Extra information includes status for user input, such as a user-input parameter or a stop for manual operation. In the case, the process status is FLOW_IN_PROGRESS and the additional information is WAITING FOR MANUAL OPERATION or USER INPUT.

A string command example:

```
>nolio.cmd get-job-status -u superuser -p suser -i -40
```

String output:

```
[Process Name] [string-based result status][additional information]
```

Note: You can query a process job status from different consoles even when the process is running in synchronous mode.

Use a Configuration File with Release Operations Center

The use of a configuration file enables updating releases as follows:

- Step name
- Step description
- Step dependencies
- Server dependencies in a specific step
- Release property
- Assigning a release property to a parameter used in the step
- Link a File Parameter to an Artifact

The CLI syntax for Release Operations Center is:

```
nolio.cmd update-release -c cli-file-path –r releaseName –v releaseVersion –a
applicationName –e environmentName -u superuser -p suser
```

Note: You can place all CLI commands in a Configuration File, however, any commands set in a Configuration File gets overridden by the values of the same commands that are also set in the CLI.

Elements of Configuration File

The following table describes the configuration file elements and attributes:

Element Tag	Attribute/Description
name	Release name
description	Release description.
type	Release type <minor><major><emergency>
version	Release version
properties	Release properties
steps	Step name

Element Tag	Attribute/Description
parameters	name="[folder/sub-folder/server-type/parameter-name]" value="[parameter-value]" The param attribute can be used under the app-param and server elements.
artifacts	Artifact name and version

Example of an XML Configuration File

The following example is of a configuration file that is used for Release Operation Center:

```
<release>
  <!-- general release details. all are optional-->
  <name>New release name</name>
  <description>New description</description>
  <type>minor</type>
  04.7.001.0.5</version>
  <properties>
    <property name="prop-1">property value</property>
    <property name="prop-2">property value</property>
    </properties>
    <steps>
     <step name="my step name">
      <name>new name of the step</name>
      04.7.001.5.5 - new version</version>
      <server-type name="st1">
        <!-- the servers that are running this step-->
        <servers>
          <server name="first-agent"/>
          <server name="second-agent" >
            <dependencies>
              <dependency server-type="st2" server="third-agent"/>
            </dependencies>
          </server>
          </servers>
          <parameters>
            <parameter name="folder/sub-folder/server-type parameter to set">
              <!-- each entry can have value "fromProperty" - which should
                   be linked to an existing release property, or "value" -
                   which is a regular string.
                   in case of arrays, each entry in the array should have
                   a "value" tag.
                   "default" tag can appear at most one time, server can be at most
                   the number of the servers associated with this server type.
                   Global parameters will be set outside the server type scope.
                -->
              <default fromProperty="prop-1"/>
              <server name="first-agent" fromProperty="prop-1"/>
              <server name="second-agent" value="my value"/>
            </parameter>
            <parameter name="folder/sub-folder/server-type array parameter">
              <server name="second-agent">
                <value value="v1" />
                <value value="v2" />
                <value value="v3" />
             </server>
```

```
            </parameter>
          </parameters>
          <artifacts>
            <parameter name="file-parameter1">
              <!-- single artifact per each file parameter. name and version are
required -->
              <artifact name="Arti1" version="1.0.0"/>
            </parameter>
            <parameter name="file-parameter2">
              <artifact name="Arti2" version="1.0.0"/>
            </parameter>
          </artifacts>
        </server-type>
        <server-type name="st2">
          <!-- the servers that are running this step-->
          <servers>
            <server name="third-agent"/>
          </servers>
        </server-type>
        <!-- global parameters -->
        <parameters>
          <parameter name="process parameter" fromProperty="prop-2"/>
          <parameter name="Application Parameters/my app parameter" value="app
param value" />
          <parameter name="Application Parameters/global parameter array">
            <value value="v1" />
            <value value="v2" />
            <value value="v3" />
          </parameter>
        </parameters>
      </step>
      <step name="second step">
        <!-- set the dependencies between steps -->
        <dependencies>
          <step name="my lovely step"/>
          <!-- you can specify how many steps you want. for example:
          <step name="s3"/>
          -->
        </dependencies>
      </step>
    </steps>
</release>
```

Rules for Using an XML Configuration File

The following rules apply to using an XML Configuration File in the CLI:

- If you use a Configuration File, you cannot use the parameter (-r) and server (-s) switches in the CLI.
- A command that is entered in the CLI overrides the same command that is entered in the Configuration File.

For example, you enter '-a' in the CLI and have a tag for 'application' in the Configuration file, the CLI '-a' command overrides the parameter data that is contained in the Configuration File tag.

- The specified Configuration File must reside on the computer where the CLI is executed and not on a Management server.
- Server Type names must be unique per application.
- Server name can be either its IP address or its Node ID.
- For each server, you can optionally define dependencies in the Configuration File. You cannot define dependencies in the CLI.

Server dependencies are added or changed but not deleted from the Environment using the Configuration File.

- Commas can be included in the parameter value without the backslash as is required for the CLI.

Use a Configuration File with Automation Studio

You cannot modify parameters and server dependencies for specific servers executing under multiple server types using the CLI. However, it is possible to do so by using an XML configuration file in the CLI.

The CLI syntax for Automation Studio is:

```
nolio.cmd run-release -u superuser -p suser -a applicationName -e environmentName -f
processName -c cli-file-path
```

Note: You can place all CLI commands in a configuration file, however, any command set in a configuration file gets overridden by the same command that is also set in the CLI.

Elements of Configuration File

The following table describes the Configuration File elements and attributes:

Element Tag	Attribute/Description
app-param	Application level parameters definition block Parameters set in the app-param block apply to the entire application.
application	name="[application-name]"
nolio	Required root element
dependencies	Dependencies definition block [optional]
dependency	server-type="[server-type-name]" server="[ip-address]" [optional]
env	name="[environment-name]"
param	name="[folder/parameter-name]" value="[parameter-value]" The param attribute can be used under the app-param and server elements.
value	name="[array-parameter-value-1]" ... name="[array-parameter-value-n]" The value attribute can be used to set global arrays or server type arrays.
parameters	Parameters definition block
process	name="[process-name]"
server	name="[ip-address or node-id]" Parameters set under the server element apply only to that server.
server-types	Server Types definition block
server-type	name="[server-type-name]"
user	name="[user-name]"

Example of an XML Configuration File

The following example is of a configuration file for Automation Studio:

```
<nolio>
  <application name="MyApplication" />
  <env name="MyEnvironment" />
  <process name="MyProcess" />
  <user name="superuser" />
  <server-types>
    <server-type name="Server Type 1">
     <server name="127.0.0.1" />
      <server name="iceman2" />
    </server-type>
    <server-type name="Server Type 2">
      <server name="MyServer">
        <dependencies>
          <dependency server-type="Server Type 1" server="127.0.0.1" />
        </dependencies>
      </server>
    </server-type>
    <server-type name="e">
      <server name="MyServer">
        <dependencies>
          <dependency server-type="Server Type 2" server="127.0.0.1" />
        </dependencies>
      </server>
    </server-type>
  </server-types>
  <parameters>
    <app-param>
      <param name="Application Parameters/a" value="global parameter value"/>
    </app-param>
    <server-type name="Server Type 1">
      <server name="127.0.0.1">
        <param name="f/ui" value="value on ST1" />
      </server>
    </server-type>
    <server-type name="Server Type 2">
      <server name="127.0.0.1">
        <param name="f/ui" value="value on ST2" />
      </server>
    </server-type>
  </parameters>
</nolio>
```

Example of Array Value Syntax

The following example is of the array value syntax:

```
<app-param>\
    <!—regular parameter-->
    <param name="Folder_Name/Parameter_Name" value="Global_Parameter_Value" />
    <!—array parameter-->
        <param name="Folder_Name/Array_Parameter_Name2">
            <value>Global_Array_Parameter_Value_1</value>
            <value>Global_Array_Parameter_Value_2</value>
            <value>Global_Array_Parameter_Value_3</value>
        </param>
</app-param>
```

Scope of Configuration File Definitions

Parameter and dependency definitions in a Configuration File have different effects on subsequent runs:

- Parameters that are defined in the Configuration File are only active for the current CLI execution.
- Dependencies that are defined in the Configuration File are active for the current and future executions, because the CA Release Automation database definition is updated with the Configuration File value.

When the dependencies are defined in the Configuration File, the definitions override the database environment values and are active beyond the CLI execution.

When the dependencies are not defined in the Configuration File, Environment values are used.

View Execution and System Logs

The following logs for CLI users are also available in CA Release Automation.

- Execution-oriented logs: *CAAgent\logs\nolio_action_exe.log*
- System detail logs: *CAAgent\logs\nolio_all.log*
- High-level logs: *CACLI Installation folder nolio_app_all.log*

Chapter 3: SOAP API Reference

CA Release Automation offers an Open API which is a SOAP Web Service that is designed to enable third-party integration with Automation Studio. The SOAP API enables the configuration, execution, and monitoring of processes without having to access the Release Automation GUI.

Open API and Web Services

The Open API and Web services allow:

- Retrieval of Release Automation applications, environments, published processes, and agents.
- Retrieval of the required user input parameters for a process.
- The ability to configure and execute processes.
- Query of status, errors, and failed actions of a process.
- The ability to start and stop a process.
- Schedule a single process run.
- Remove all or specific agents from an environment.
- Map an agent to an environment.
- Create an export file of application data.

The Automation Studio Web Service Definition Language (WSDL) is available at:

http://<host>:<port>/datamanagement/ws

For more information about the Open API Web Service, refer to the Javadoc documentation available in CA Release Automation.

```
<NAC Root>//datamanagement/docs/index.html
http://<host>:< port>/datamanagement/docs/index.html
```

For more information on the SOAP Return Codes, see SOAP Return Codes (see page 94)

The SOAP and Open API Method List

The available CA Release Automation SOAP methods that are located in the ExecutionRelayWS interface are:

runProcess

Sets up a process run.

runProcessAsync

Run asynchronously (does not wait until execute terminates).

getProcessStatusAsString

Returns the process status as a string.

getProcessStatusAsInt

Returns the process status as an integer.

For more information see, ExecutionRelayWS (see page 37)

The available CA Release Automation Open API methods that are located in the Open APIService interface are:

assignTagToEnvironment

Assigns tagged processes to an environment.

createEnvironment

Creates new environment in an application

exportApplications

Creates an export file that contains application data

getAllAgents

Retrieves all known agents.

getAllApplications

Gets all applications in the system.

getAssignedProcessesForEnvironment

Gets all processes that are assigned to an environment.

getConnectedAgentsForES

Returns a list of agents connected to an execution server.

getConnectedESForAgent

Returns the execution server assigned to an agent.

getEnvironmentsForApplication

Gets all environments in an application.

getEnvironmentServers

Gets all servers in an environment.

getJobErrors

Retrieves the current errors in a run.

getJobFailedSteps

Retrieves the current failed actions in a run

getJobStatus

Queries the status of a run.

getProcessStatusAsString

Returns the process status as a string.

getProcessStatusAsInt

Returns the process status as an integer.

getUserInputParameters

Returns the unassigned parameters that are required to process a run.

mapInstanceToEnvironment

Maps an agent instance to an environment.

removeAllAgentsFromEnvironment

Removes all agents from an environment.

removeInstanceFromEnvironment

Removes an agent from an environment

removeTagFromEnvironment

Removes tagged processes from an environment.

runProcess

Sets up a process run.

runProcess2

Sets up a process run according to a configuration.

Note: To reference a process tag, the full existing tag name is required.

runProcessAsync

Run asynchronously (does not wait until execute terminates).

scheduleProcessRun

Schedules a single process run according to a configuration.

stopJob

Stops an active job.

For more information see, Open API Service (see page 44)

ExecutionRelayWS

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

- <wsdl:definitions name="ExecutionRelayWs"
targetNamespace="http://execution.api.dataservices.server.platform.nolio.com/"
xmlns:ns1="http://schemas.xmlsoap.org/soap/http"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:tns="http://execution.api.dataservices.server.platform.nolio.com/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">

- <wsdl:types>

- <xsd:schema attributeFormDefault="qualified" elementFormDefault="qualified"
targetNamespace="http://execution.api.dataservices.server.platform.nolio.com/"
xmlns:tns="http://execution.api.dataservices.server.platform.nolio.com/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">

- <xsd:complexType name="ArrayOfString">

- <xsd:sequence>

<xsd:element maxOccurs="unbounded" minOccurs="0" name="string" nillable="true"
type="xsd:string" />

</xsd:sequence>

</xsd:complexType>

- <xsd:complexType name="string2stringMap">

+ <xsd:sequence>

- <xsd:element maxOccurs="unbounded" minOccurs="0" name="entry">

- <xsd:complexType>

- <xsd:sequence>

<xsd:element name="key" type="xsd:string" />

<xsd:element minOccurs="0" name="value" type="xsd:string" />

</xsd:sequence>

</xsd:complexType>

</xsd:element>

</xsd:sequence>

</xsd:complexType>
```

```
<xsd:element name="runProcess" type="tns:runProcess" />

+ <xsd:complexType name="runProcess">

- <xsd:sequence>

<xsd:element minOccurs="0" name="username" type="xsd:string" />

<xsd:element minOccurs="0" name="password" type="xsd:string" />

<xsd:element minOccurs="0" name="appName" type="xsd:string" />

<xsd:element minOccurs="0" name="processName" type="xsd:string" />

<xsd:element minOccurs="0" name="environmentName" type="xsd:string" />

<xsd:element minOccurs="0" name="servers" type="tns:ArrayOfString" />

<xsd:element minOccurs="0" name="parameters" nillable="true"
type="tns:string2stringMap" />

<xsd:element name="timeout" type="xsd:int" />

<xsd:element minOccurs="0" name="processTag" nillable="true" type="xsd:string" />

</xsd:sequence>

</xsd:complexType>

<xsd:element name="runProcessResponse" type="tns:runProcessResponse" />

- <xsd:complexType name="runProcessResponse">

- <xsd:sequence>

<xsd:element minOccurs="0" name="return" nillable="true" type="xsd:long" />

</xsd:sequence>

</xsd:complexType>

<xsd:element name="runProcessAsync" type="tns:runProcessAsync" />

- <xsd:complexType name="runProcessAsync">

- <xsd:sequence>

<xsd:element minOccurs="0" name="username" type="xsd:string" />

<xsd:element minOccurs="0" name="password" type="xsd:string" />

<xsd:element minOccurs="0" name="appName" type="xsd:string" />

<xsd:element minOccurs="0" name="processName" type="xsd:string" />

<xsd:element minOccurs="0" name="environmentName" type="xsd:string" />
```

```
<xsd:element minOccurs="0" name="servers" type="tns:ArrayOfString" />

<xsd:element minOccurs="0" name="parameters" nillable="true"
type="tns:string2stringMap" />

<xsd:element name="timeout" type="xsd:int" />

<xsd:element minOccurs="0" name="processTag" nillable="true" type="xsd:string" />

</xsd:sequence>

</xsd:complexType>

<xsd:element name="runProcessAsyncResponse" type="tns:runProcessAsyncResponse" />

- <xsd:complexType name="runProcessAsyncResponse">

- <xsd:sequence>

<xsd:element minOccurs="0" name="return" nillable="true" type="xsd:long" />

</xsd:sequence>

</xsd:complexType>

<xsd:element name="getProcessStatusAsString" type="tns:getProcessStatusAsString"
/>

- <xsd:complexType name="getProcessStatusAsString">

- <xsd:sequence>

<xsd:element minOccurs="0" name="processId" type="xsd:long" />

</xsd:sequence>

</xsd:complexType>

<xsd:element name="getProcessStatusAsStringResponse"
type="tns:getProcessStatusAsStringResponse" />

- <xsd:complexType name="getProcessStatusAsStringResponse">

- <xsd:sequence>

<xsd:element minOccurs="0" name="return" nillable="true" type="xsd:string" />

</xsd:sequence>

</xsd:complexType>

<xsd:element name="getProcessStatusAsInt" type="tns:getProcessStatusAsInt" />

- <xsd:complexType name="getProcessStatusAsInt">

- <xsd:sequence>
```

```
<xsd:element minOccurs="0" name="processId" type="xsd:long" />

</xsd:sequence>

</xsd:complexType>

<xsd:element name="getProcessStatusAsIntResponse"
type="tns:getProcessStatusAsIntResponse" />

- <xsd:complexType name="getProcessStatusAsIntResponse">

- <xsd:sequence>

<xsd:element name="return" type="xsd:int" />

</xsd:sequence>

</xsd:complexType>

</xsd:schema>

</wsdl:types>

- <wsdl:message name="getProcessStatusAsIntResponse">

<wsdl:part element="tns:getProcessStatusAsIntResponse" name="parameters" />

</wsdl:message>

- <wsdl:message name="getProcessStatusAsString">

<wsdl:part element="tns:getProcessStatusAsString" name="parameters" />

</wsdl:message>

- <wsdl:message name="runProcessAsync">

<wsdl:part element="tns:runProcessAsync" name="parameters" />

</wsdl:message>

- <wsdl:message name="runProcessAsyncResponse">

<wsdl:part element="tns:runProcessAsyncResponse" name="parameters" />

</wsdl:message>

- <wsdl:message name="getProcessStatusAsInt">

<wsdl:part element="tns:getProcessStatusAsInt" name="parameters" />

</wsdl:message>

- <wsdl:message name="runProcessResponse">

<wsdl:part element="tns:runProcessResponse" name="parameters" />
```

```
</wsdl:message>

- <wsdl:message name="getProcessStatusAsStringResponse">

<wsdl:part element="tns:getProcessStatusAsStringResponse" name="parameters" />

</wsdl:message>

- <wsdl:message name="runProcess">

<wsdl:part element="tns:runProcess" name="parameters" />

</wsdl:message>

- <wsdl:portType name="ExecutionRelayWsPortType">

- <wsdl:operation name="runProcess">

<wsdl:input message="tns:runProcess" name="runProcess" />

<wsdl:output message="tns:runProcessResponse" name="runProcessResponse" />

</wsdl:operation>

- <wsdl:operation name="runProcessAsync">

<wsdl:input message="tns:runProcessAsync" name="runProcessAsync" />

<wsdl:output message="tns:runProcessAsyncResponse" name="runProcessAsyncResponse"
/>

</wsdl:operation>

- <wsdl:operation name="getProcessStatusAsString">

<wsdl:input message="tns:getProcessStatusAsString" name="getProcessStatusAsString"
/>

<wsdl:output message="tns:getProcessStatusAsStringResponse"
name="getProcessStatusAsStringResponse" />

</wsdl:operation>

- <wsdl:operation name="getProcessStatusAsInt">

<wsdl:input message="tns:getProcessStatusAsInt" name="getProcessStatusAsInt" />

<wsdl:output message="tns:getProcessStatusAsIntResponse"
name="getProcessStatusAsIntResponse" />

</wsdl:operation>

</wsdl:portType>

- <wsdl:binding name="ExecutionRelayWsSoapBinding"
type="tns:ExecutionRelayWsPortType">
```

```
<soap12:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"
/>

- <wsdl:operation name="runProcess">

<soap12:operation soapAction="" style="document" />

- <wsdl:input name="runProcess">

<soap12:body use="literal" />

</wsdl:input>

- <wsdl:output name="runProcessResponse">

<soap12:body use="literal" />

</wsdl:output>

</wsdl:operation>

- <wsdl:operation name="runProcessAsync">

<soap12:operation soapAction="" style="document" />

- <wsdl:input name="runProcessAsync">

<soap12:body use="literal" />

</wsdl:input>

- <wsdl:output name="runProcessAsyncResponse">

<soap12:body use="literal" />

</wsdl:output>

</wsdl:operation>

- <wsdl:operation name="getProcessStatusAsInt">

<soap12:operation soapAction="" style="document" />

- <wsdl:input name="getProcessStatusAsInt">

<soap12:body use="literal" />

</wsdl:input>

- <wsdl:output name="getProcessStatusAsIntResponse">

<soap12:body use="literal" />

</wsdl:output>

</wsdl:operation>
```

```
- <wsdl:operation name="getProcessStatusAsString">

<soap12:operation soapAction="" style="document" />

- <wsdl:input name="getProcessStatusAsString">

<soap12:body use="literal" />

</wsdl:input>

- <wsdl:output name="getProcessStatusAsStringResponse">

<soap12:body use="literal" />

</wsdl:output>

</wsdl:operation>

</wsdl:binding>

- <wsdl:service name="ExecutionRelayWs">

- <wsdl:port binding="tns:ExecutionRelayWsSoapBinding"
name="ExecutionRelayWsPort">

<soap12:address
location="http://tamme012233:8080/datamanagement/ws/ExecutionRelayWS" />

</wsdl:port>

</wsdl:service>

</wsdl:definitions>
```

OpenAPIService

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

- <wsdl:definitions name="OpenAPIService"
targetNamespace="http://model.api.dataservices.server.platform.nolio.com/"
xmlns:ns1="http://schemas.xmlsoap.org/soap/http"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:tns="http://model.api.dataservices.server.platform.nolio.com/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">

- <wsdl:types>

- <xsd:schema attributeFormDefault="qualified" elementFormDefault="qualified"
targetNamespace="http://dto.webservice.model.api.dataservices.server.platform.nol
io.com" xmlns:ns0="http://model.api.dataservices.server.platform.nolio.com/"
xmlns:tns="http://dto.webservice.model.api.dataservices.server.platform.nolio.com
" xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:import
namespace="http://model.api.dataservices.server.platform.nolio.com/" />

- <xsd:complexType name="AgentInstanceStatusWS">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="agentInstance" nillable="true"
type="tns:AgentInstanceWS" />

  <xsd:element minOccurs="0" name="agentProgress" type="xsd:double" />

  <xsd:element minOccurs="0" name="currentStep" nillable="true" type="xsd:string"
/>

  <xsd:element minOccurs="0" name="executionState" nillable="true"
type="xsd:string" />

  </xsd:sequence>

  </xsd:complexType>

- <xsd:complexType name="ArrayOfEnvironmentWS">

- <xsd:sequence>

  <xsd:element maxOccurs="unbounded" minOccurs="0" name="EnvironmentWS"
nillable="true" type="tns:EnvironmentWS" />

  </xsd:sequence>

  </xsd:complexType>

- <xsd:complexType name="ArrayOfAgentInstancesDependencyWS">
```

```
- <xsd:sequence>

  <xsd:element maxOccurs="unbounded" minOccurs="0"
name="AgentInstancesDependencyWS" nillable="true"
type="tns:AgentInstancesDependencyWS" />

  </xsd:sequence>

  </xsd:complexType>

- <xsd:complexType name="ArrayOfParameterWS">

- <xsd:sequence>

  <xsd:element maxOccurs="unbounded" minOccurs="0" name="ParameterWS"
nillable="true" type="tns:ParameterWS" />

  </xsd:sequence>

  </xsd:complexType>

- <xsd:complexType name="ArrayOfApplicationWS">

- <xsd:sequence>

  <xsd:element maxOccurs="unbounded" minOccurs="0" name="ApplicationWS"
nillable="true" type="tns:ApplicationWS" />

  </xsd:sequence>

  </xsd:complexType>

- <xsd:complexType name="JobStatusWS">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="agentsStatuses" nillable="true"
type="tns:ArrayOfAgentInstanceStatusWS" />

  <xsd:element minOccurs="0" name="jobState" nillable="true" type="xsd:string" />

  </xsd:sequence>

  </xsd:complexType>

- <xsd:complexType name="FailedStepWS">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="agentIP" nillable="true" type="xsd:string" />

  <xsd:element minOccurs="0" name="result" nillable="true" type="xsd:string" />

  <xsd:element minOccurs="0" name="startTimeUTC" type="xsd:long" />

  <xsd:element minOccurs="0" name="stepTitle" nillable="true" type="xsd:string" />
```

```
  </xsd:sequence>

  </xsd:complexType>

- <xsd:complexType name="AgentWS">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="description" nillable="true" type="xsd:string"
/>

  <xsd:element minOccurs="0" name="hostName" nillable="true" type="xsd:string" />

  <xsd:element minOccurs="0" name="ip" nillable="true" type="xsd:string" />

  <xsd:element minOccurs="0" name="port" type="xsd:int" />

  <xsd:element minOccurs="0" name="reachable" type="xsd:boolean" />

  </xsd:sequence>

  </xsd:complexType>

- <xsd:complexType name="AgentInstanceWS">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="agentHostOrIp" nillable="true" type="xsd:string"
/>

  <xsd:element minOccurs="0" name="serverTypeName" nillable="true"
type="xsd:string" />

  </xsd:sequence>

  </xsd:complexType>

- <xsd:complexType name="ArrayOfEnvironmentServerWS">

- <xsd:sequence>

  <xsd:element maxOccurs="unbounded" minOccurs="0" name="EnvironmentServerWS"
nillable="true" type="tns:EnvironmentServerWS" />

  </xsd:sequence>

  </xsd:complexType>

- <xsd:complexType name="ArrayOfExecutionServerWS">

- <xsd:sequence>

  <xsd:element maxOccurs="unbounded" minOccurs="0" name="ExecutionServerWS"
nillable="true" type="tns:ExecutionServerWS" />

  </xsd:sequence>
```

```
  </xsd:complexType>
- <xsd:complexType name="ExecutionServerWS">
- <xsd:sequence>
  <xsd:element minOccurs="0" name="hostName" nillable="true" type="xsd:string" />
  <xsd:element minOccurs="0" name="nodeId" nillable="true" type="xsd:string" />
  <xsd:element minOccurs="0" name="port" type="xsd:int" />
  <xsd:element minOccurs="0" name="reachable" type="xsd:boolean" />
  </xsd:sequence>
  </xsd:complexType>
- <xsd:complexType name="ProcessWS">
- <xsd:sequence>
  <xsd:element minOccurs="0" name="description" nillable="true" type="xsd:string"
/>
  <xsd:element minOccurs="0" name="id" type="xsd:long" />
  <xsd:element minOccurs="0" name="latest" nillable="true" type="xsd:boolean" />
  <xsd:element minOccurs="0" name="processFullName" nillable="true"
type="xsd:string" />
  <xsd:element minOccurs="0" name="serverTypes" nillable="true"
type="ns0:ArrayOfString" />
  <xsd:element minOccurs="0" name="tagDescription" nillable="true"
type="xsd:string" />
  <xsd:element minOccurs="0" name="tagName" nillable="true" type="xsd:string" />
  </xsd:sequence>
  </xsd:complexType>
- <xsd:complexType name="ArrayOfProcessWS">
- <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="ProcessWS" nillable="true"
type="tns:ProcessWS" />
  </xsd:sequence>
  </xsd:complexType>
- <xsd:complexType name="EnvironmentWS">
```

```
- <xsd:sequence>

  <xsd:element minOccurs="0" name="description" nillable="true" type="xsd:string"
/>

  <xsd:element minOccurs="0" name="id" type="xsd:long" />

  <xsd:element minOccurs="0" name="name" nillable="true" type="xsd:string" />

  </xsd:sequence>

  </xsd:complexType>

- <xsd:complexType name="ArrayOfFailedStepWS">

- <xsd:sequence>

  <xsd:element maxOccurs="unbounded" minOccurs="0" name="FailedStepWS"
nillable="true" type="tns:FailedStepWS" />

  </xsd:sequence>

  </xsd:complexType>

- <xsd:complexType name="ArrayOfAgentWS">

- <xsd:sequence>

  <xsd:element maxOccurs="unbounded" minOccurs="0" name="AgentWS" nillable="true"
type="tns:AgentWS" />

  </xsd:sequence>

  </xsd:complexType>

- <xsd:complexType name="ParameterAssignmentWS">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="container" nillable="true"
type="tns:AgentInstanceWS" />

  <xsd:element minOccurs="0" name="parameterPathName" nillable="true"
type="xsd:string" />

  <xsd:element minOccurs="0" name="value" nillable="true" type="xsd:string" />

  </xsd:sequence>

  </xsd:complexType>

- <xsd:complexType name="AgentInstancesDependencyWS">

- <xsd:sequence>
```

```
  <xsd:element minOccurs="0" name="source" nillable="true"
type="tns:AgentInstanceWS" />

  <xsd:element minOccurs="0" name="target" nillable="true"
type="tns:AgentInstanceWS" />

  </xsd:sequence>

  </xsd:complexType>

- <xsd:complexType name="ArrayOfAgentInstanceWS">

- <xsd:sequence>

  <xsd:element maxOccurs="unbounded" minOccurs="0" name="AgentInstanceWS"
nillable="true" type="tns:AgentInstanceWS" />

  </xsd:sequence>

  </xsd:complexType>

- <xsd:complexType name="ParameterWS">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="parameterPathName" nillable="true"
type="xsd:string" />

  <xsd:element minOccurs="0" name="serverTypeName" nillable="true"
type="xsd:string" />

  </xsd:sequence>

  </xsd:complexType>

- <xsd:complexType name="ApplicationWS">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="description" nillable="true" type="xsd:string"
/>

  <xsd:element minOccurs="0" name="id" type="xsd:long" />

  <xsd:element minOccurs="0" name="name" nillable="true" type="xsd:string" />

  </xsd:sequence>

  </xsd:complexType>

- <xsd:complexType name="ArrayOfAgentInstanceStatusWS">

- <xsd:sequence>
```

```
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="AgentInstanceStatusWS"
nillable="true" type="tns:AgentInstanceStatusWS" />

  </xsd:sequence>

  </xsd:complexType>

- <xsd:complexType name="EnvironmentServerWS">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="mappedServerHostName" nillable="true"
type="xsd:string" />

  <xsd:element minOccurs="0" name="mappedServerIp" nillable="true"
type="xsd:string" />

  <xsd:element minOccurs="0" name="serverTypeName" nillable="true"
type="xsd:string" />

  </xsd:sequence>

  </xsd:complexType>

- <xsd:complexType name="ProcessRunConfigurationWS">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="agentInstances" nillable="true"
type="tns:ArrayOfAgentInstanceWS" />

  <xsd:element minOccurs="0" name="applicationName" nillable="true"
type="xsd:string" />

  <xsd:element minOccurs="0" name="dependencies" nillable="true"
type="tns:ArrayOfAgentInstancesDependencyWS" />

  <xsd:element minOccurs="0" name="environmentName" nillable="true"
type="xsd:string" />

  <xsd:element minOccurs="0" name="parameters" nillable="true"
type="tns:ArrayOfParameterAssignmentWS" />

  <xsd:element minOccurs="0" name="processFullName" nillable="true"
type="xsd:string" />

  <xsd:element minOccurs="0" name="processTag" nillable="true" type="xsd:string" />

  </xsd:sequence>

  </xsd:complexType>

- <xsd:complexType name="ArrayOfParameterAssignmentWS">

- <xsd:sequence>
```

```
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="ParameterAssignmentWS"
nillable="true" type="tns:ParameterAssignmentWS" />

  </xsd:sequence>

  </xsd:complexType>

  </xsd:schema>

- <xsd:schema attributeFormDefault="qualified" elementFormDefault="qualified"
targetNamespace="http://model.api.dataservices.server.platform.nolio.com/"
xmlns:ns0="http://dto.webservice.model.api.dataservices.server.platform.nolio.com
"
xmlns:ns1="http://exceptions.webservice.model.api.dataservices.server.platform.no
lio.com"
xmlns:ns2="http://notifications.rc.api.dataservices.server.platform.nolio.com"
xmlns:tns="http://model.api.dataservices.server.platform.nolio.com/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:import
namespace="http://dto.webservice.model.api.dataservices.server.platform.nolio.com
" />

- <xsd:complexType name="string2stringMap">

- <xsd:sequence>

- <xsd:element maxOccurs="unbounded" minOccurs="0" name="entry">

- <xsd:complexType>

- <xsd:sequence>

  <xsd:element name="key" type="xsd:string" />

  <xsd:element minOccurs="0" name="value" type="xsd:string" />

  </xsd:sequence>

  </xsd:complexType>

  </xsd:element>

  </xsd:sequence>

  </xsd:complexType>

- <xsd:complexType name="ArrayOfString">

- <xsd:sequence>

  <xsd:element maxOccurs="unbounded" minOccurs="0" name="string" nillable="true"
type="xsd:string" />

  </xsd:sequence>
```

```
  </xsd:complexType>

  <xsd:element name="removeInstanceFromEnvironment"
type="tns:removeInstanceFromEnvironment" />

- <xsd:complexType name="removeInstanceFromEnvironment">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="userName" type="xsd:string" />

  <xsd:element minOccurs="0" name="password" type="xsd:string" />

  <xsd:element minOccurs="0" name="agent" type="xsd:string" />

  <xsd:element minOccurs="0" name="applicationName" type="xsd:string" />

  <xsd:element minOccurs="0" name="environment" type="xsd:string" />

  <xsd:element minOccurs="0" name="serverTypeName" type="xsd:string" />

  </xsd:sequence>

  </xsd:complexType>

  <xsd:element name="removeInstanceFromEnvironmentResponse"
type="tns:removeInstanceFromEnvironmentResponse" />

- <xsd:complexType name="removeInstanceFromEnvironmentResponse">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="return" nillable="true" type="xsd:long" />

  </xsd:sequence>

  </xsd:complexType>

  <xsd:element name="getAllAgents" type="tns:getAllAgents" />

- <xsd:complexType name="getAllAgents">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="username" type="xsd:string" />

  <xsd:element minOccurs="0" name="password" type="xsd:string" />

  </xsd:sequence>

  </xsd:complexType>

  <xsd:element name="getAllAgentsResponse" type="tns:getAllAgentsResponse" />

- <xsd:complexType name="getAllAgentsResponse">
```

```
- <xsd:sequence>
  <xsd:element name="return" nillable="true" type="ns0:ArrayOfAgentWS" />
  </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="getConnectedAgentsForES" type="tns:getConnectedAgentsForES"
/>
- <xsd:complexType name="getConnectedAgentsForES">
- <xsd:sequence>
  <xsd:element minOccurs="0" name="username" type="xsd:string" />
  <xsd:element minOccurs="0" name="password" type="xsd:string" />
  <xsd:element minOccurs="0" name="executionServerNodeId" type="xsd:string" />
  </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="getConnectedAgentsForESResponse"
type="tns:getConnectedAgentsForESResponse" />
- <xsd:complexType name="getConnectedAgentsForESResponse">
- <xsd:sequence>
  <xsd:element name="return" nillable="true" type="ns0:ArrayOfAgentWS" />
  </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="assignTagToEnvironment" type="tns:assignTagToEnvironment" />
- <xsd:complexType name="assignTagToEnvironment">
- <xsd:sequence>
  <xsd:element minOccurs="0" name="username" type="xsd:string" />
  <xsd:element minOccurs="0" name="password" type="xsd:string" />
  <xsd:element minOccurs="0" name="appName" type="xsd:string" />
  <xsd:element minOccurs="0" name="envName" type="xsd:string" />
  <xsd:element minOccurs="0" name="procFullName" type="xsd:string" />
  <xsd:element minOccurs="0" name="tagName" type="xsd:string" />
```

```
  </xsd:sequence>

  </xsd:complexType>

  <xsd:element name="assignTagToEnvironmentResponse"
type="tns:assignTagToEnvironmentResponse" />

- <xsd:complexType name="assignTagToEnvironmentResponse">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="return" nillable="true" type="xsd:long" />

  </xsd:sequence>

  </xsd:complexType>

  <xsd:element name="getJobFailedSteps" type="tns:getJobFailedSteps" />

- <xsd:complexType name="getJobFailedSteps">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="username" type="xsd:string" />

  <xsd:element minOccurs="0" name="password" type="xsd:string" />

  <xsd:element name="jobId" type="xsd:long" />

  </xsd:sequence>

  </xsd:complexType>

  <xsd:element name="getJobFailedStepsResponse"
type="tns:getJobFailedStepsResponse" />

- <xsd:complexType name="getJobFailedStepsResponse">

- <xsd:sequence>

  <xsd:element name="return" nillable="true" type="ns0:ArrayOfFailedStepWS" />

  </xsd:sequence>

  </xsd:complexType>

  <xsd:element name="getConnectedESForAgent" type="tns:getConnectedESForAgent" />

- <xsd:complexType name="getConnectedESForAgent">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="username" type="xsd:string" />

  <xsd:element minOccurs="0" name="password" type="xsd:string" />
```

```
  <xsd:element minOccurs="0" name="agentNodeId" type="xsd:string" />

  </xsd:sequence>

  </xsd:complexType>

  <xsd:element name="getConnectedESForAgentResponse"
type="tns:getConnectedESForAgentResponse" />

- <xsd:complexType name="getConnectedESForAgentResponse">

- <xsd:sequence>

  <xsd:element name="return" nillable="true" type="ns0:ArrayOfExecutionServerWS" />

  </xsd:sequence>

  </xsd:complexType>

  <xsd:element name="removeAllAgentsFromEnvironment"
type="tns:removeAllAgentsFromEnvironment" />

- <xsd:complexType name="removeAllAgentsFromEnvironment">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="userName" type="xsd:string" />

  <xsd:element minOccurs="0" name="password" type="xsd:string" />

  <xsd:element minOccurs="0" name="applicationName" type="xsd:string" />

  <xsd:element minOccurs="0" name="environment" type="xsd:string" />

  </xsd:sequence>

  </xsd:complexType>

  <xsd:element name="removeAllAgentsFromEnvironmentResponse"
type="tns:removeAllAgentsFromEnvironmentResponse" />

- <xsd:complexType name="removeAllAgentsFromEnvironmentResponse">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="return" nillable="true" type="xsd:long" />

  </xsd:sequence>

  </xsd:complexType>

  <xsd:element name="getJobErrors" type="tns:getJobErrors" />

- <xsd:complexType name="getJobErrors">

- <xsd:sequence>
```

```
  <xsd:element minOccurs="0" name="username" type="xsd:string" />
  <xsd:element minOccurs="0" name="password" type="xsd:string" />
  <xsd:element name="jobId" type="xsd:long" />
  </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="getJobErrorsResponse" type="tns:getJobErrorsResponse" />
- <xsd:complexType name="getJobErrorsResponse">
- <xsd:sequence>
  <xsd:element name="return" nillable="true" type="tns:ArrayOfString" />
  </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="exportApplications" type="tns:exportApplications" />
- <xsd:complexType name="exportApplications">
- <xsd:sequence>
  <xsd:element minOccurs="0" name="username" type="xsd:string" />
  <xsd:element minOccurs="0" name="password" type="xsd:string" />
  <xsd:element name="appNameList" nillable="true" type="tns:ArrayOfString" />
  <xsd:element minOccurs="0" name="filePath" type="xsd:string" />
  </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="exportApplicationsResponse"
type="tns:exportApplicationsResponse" />
- <xsd:complexType name="exportApplicationsResponse">
- <xsd:sequence>
  <xsd:element minOccurs="0" name="return" nillable="true" type="xsd:string" />
  </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="getJobStatus" type="tns:getJobStatus" />
- <xsd:complexType name="getJobStatus">
```

```
- <xsd:sequence>
  <xsd:element minOccurs="0" name="username" type="xsd:string" />
  <xsd:element minOccurs="0" name="password" type="xsd:string" />
  <xsd:element name="jobId" type="xsd:long" />
  </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="getJobStatusResponse" type="tns:getJobStatusResponse" />
- <xsd:complexType name="getJobStatusResponse">
- <xsd:sequence>
  <xsd:element minOccurs="0" name="return" type="ns0:JobStatusWS" />
  </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="scheduleProcessRun" type="tns:scheduleProcessRun" />
- <xsd:complexType name="scheduleProcessRun">
- <xsd:sequence>
  <xsd:element minOccurs="0" name="username" type="xsd:string" />
  <xsd:element minOccurs="0" name="password" type="xsd:string" />
  <xsd:element minOccurs="0" name="scheduleName" type="xsd:string" />
  <xsd:element minOccurs="0" name="config" type="ns0:ProcessRunConfigurationWS" />
  <xsd:element name="startTimeUTC" type="xsd:long" />
  </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="scheduleProcessRunResponse"
type="tns:scheduleProcessRunResponse" />
- <xsd:complexType name="scheduleProcessRunResponse">
- <xsd:sequence>
  <xsd:element minOccurs="0" name="return" nillable="true" type="xsd:long" />
  </xsd:sequence>
  </xsd:complexType>
```

```
  <xsd:element name="getAssignedProcessesForEnvironment"
type="tns:getAssignedProcessesForEnvironment" />

- <xsd:complexType name="getAssignedProcessesForEnvironment">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="username" type="xsd:string" />

  <xsd:element minOccurs="0" name="password" type="xsd:string" />

  <xsd:element minOccurs="0" name="appName" type="xsd:string" />

  <xsd:element minOccurs="0" name="envName" type="xsd:string" />

  </xsd:sequence>

  </xsd:complexType>

  <xsd:element name="getAssignedProcessesForEnvironmentResponse"
type="tns:getAssignedProcessesForEnvironmentResponse" />

- <xsd:complexType name="getAssignedProcessesForEnvironmentResponse">

- <xsd:sequence>

  <xsd:element name="return" nillable="true" type="ns0:ArrayOfProcessWS" />

  </xsd:sequence>

  </xsd:complexType>

  <xsd:element name="removeTagFromEnvironment" type="tns:removeTagFromEnvironment"
/>

- <xsd:complexType name="removeTagFromEnvironment">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="username" type="xsd:string" />

  <xsd:element minOccurs="0" name="password" type="xsd:string" />

  <xsd:element minOccurs="0" name="appName" type="xsd:string" />

  <xsd:element minOccurs="0" name="envName" type="xsd:string" />

  <xsd:element minOccurs="0" name="procFullName" type="xsd:string" />

  <xsd:element minOccurs="0" name="tagName" type="xsd:string" />

  </xsd:sequence>

  </xsd:complexType>
```

```
  <xsd:element name="removeTagFromEnvironmentResponse"
type="tns:removeTagFromEnvironmentResponse" />

- <xsd:complexType name="removeTagFromEnvironmentResponse">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="return" nillable="true" type="xsd:long" />

  </xsd:sequence>

  </xsd:complexType>

  <xsd:element name="getEnvironmentServers" type="tns:getEnvironmentServers" />

- <xsd:complexType name="getEnvironmentServers">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="username" type="xsd:string" />

  <xsd:element minOccurs="0" name="password" type="xsd:string" />

  <xsd:element minOccurs="0" name="appName" type="xsd:string" />

  <xsd:element minOccurs="0" name="envName" type="xsd:string" />

  </xsd:sequence>

  </xsd:complexType>

  <xsd:element name="getEnvironmentServersResponse"
type="tns:getEnvironmentServersResponse" />

- <xsd:complexType name="getEnvironmentServersResponse">

- <xsd:sequence>

  <xsd:element name="return" nillable="true" type="ns0:ArrayOfEnvironmentServerWS"
/>

  </xsd:sequence>

  </xsd:complexType>

  <xsd:element name="getEnvironmentsForApplication"
type="tns:getEnvironmentsForApplication" />

- <xsd:complexType name="getEnvironmentsForApplication">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="username" type="xsd:string" />

  <xsd:element minOccurs="0" name="password" type="xsd:string" />
```

```
  <xsd:element minOccurs="0" name="appName" type="xsd:string" />
  </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="getEnvironmentsForApplicationResponse"
type="tns:getEnvironmentsForApplicationResponse" />
- <xsd:complexType name="getEnvironmentsForApplicationResponse">
- <xsd:sequence>
  <xsd:element name="return" nillable="true" type="ns0:ArrayOfEnvironmentWS" />
  </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="runProcess2" type="tns:runProcess2" />
- <xsd:complexType name="runProcess2">
- <xsd:sequence>
  <xsd:element minOccurs="0" name="username" type="xsd:string" />
  <xsd:element minOccurs="0" name="password" type="xsd:string" />
  <xsd:element minOccurs="0" name="config" type="ns0:ProcessRunConfigurationWS" />
  <xsd:element name="wait" type="xsd:boolean" />
  <xsd:element name="timeout" type="xsd:int" />
  <xsd:element minOccurs="0" name="jobName" type="xsd:string" />
  <xsd:element minOccurs="0" name="stop" nillable="true" type="xsd:boolean" />
  </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="runProcess2Response" type="tns:runProcess2Response" />
- <xsd:complexType name="runProcess2Response">
- <xsd:sequence>
  <xsd:element name="return" type="xsd:long" />
  </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="runProcess" type="tns:runProcess" />
```

```
- <xsd:complexType name="runProcess">
- <xsd:sequence>
  <xsd:element minOccurs="0" name="username" type="xsd:string" />
  <xsd:element minOccurs="0" name="password" type="xsd:string" />
  <xsd:element minOccurs="0" name="appName" type="xsd:string" />
  <xsd:element minOccurs="0" name="processName" type="xsd:string" />
  <xsd:element minOccurs="0" name="environmentName" type="xsd:string" />
  <xsd:element minOccurs="0" name="servers" type="tns:ArrayOfString" />
  <xsd:element minOccurs="0" name="parameters" nillable="true"
type="tns:string2stringMap" />
  <xsd:element name="timeout" type="xsd:int" />
  </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="runProcessResponse" type="tns:runProcessResponse" />
- <xsd:complexType name="runProcessResponse">
- <xsd:sequence>
  <xsd:element minOccurs="0" name="return" nillable="true" type="xsd:long" />
  </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="getUserInputParameters" type="tns:getUserInputParameters" />
- <xsd:complexType name="getUserInputParameters">
- <xsd:sequence>
  <xsd:element minOccurs="0" name="username" type="xsd:string" />
  <xsd:element minOccurs="0" name="password" type="xsd:string" />
  <xsd:element minOccurs="0" name="appName" type="xsd:string" />
  <xsd:element minOccurs="0" name="envName" type="xsd:string" />
  <xsd:element minOccurs="0" name="processFullName" type="xsd:string" />
  </xsd:sequence>
  </xsd:complexType>
```

```
  <xsd:element name="getUserInputParametersResponse"
type="tns:getUserInputParametersResponse" />

- <xsd:complexType name="getUserInputParametersResponse">

- <xsd:sequence>

  <xsd:element name="return" nillable="true" type="ns0:ArrayOfParameterWS" />

  </xsd:sequence>

  </xsd:complexType>

  <xsd:element name="stopJob" type="tns:stopJob" />

- <xsd:complexType name="stopJob">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="username" type="xsd:string" />

  <xsd:element minOccurs="0" name="password" type="xsd:string" />

  <xsd:element name="jobId" type="xsd:long" />

  </xsd:sequence>

  </xsd:complexType>

  <xsd:element name="stopJobResponse" type="tns:stopJobResponse" />

- <xsd:complexType name="stopJobResponse">

  <xsd:sequence />

  </xsd:complexType>

  <xsd:element name="mapInstanceToEnvironment" type="tns:mapInstanceToEnvironment"
/>

- <xsd:complexType name="mapInstanceToEnvironment">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="userName" type="xsd:string" />

  <xsd:element minOccurs="0" name="password" type="xsd:string" />

  <xsd:element minOccurs="0" name="agent" type="xsd:string" />

  <xsd:element minOccurs="0" name="applicationName" type="xsd:string" />

  <xsd:element minOccurs="0" name="environment" type="xsd:string" />

  <xsd:element minOccurs="0" name="serverTypeName" type="xsd:string" />
```

```
  </xsd:sequence>

  </xsd:complexType>

  <xsd:element name="mapInstanceToEnvironmentResponse"
type="tns:mapInstanceToEnvironmentResponse" />

- <xsd:complexType name="mapInstanceToEnvironmentResponse">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="return" nillable="true" type="xsd:long" />

  </xsd:sequence>

  </xsd:complexType>

  <xsd:element name="createEnvironment" type="tns:createEnvironment" />

- <xsd:complexType name="createEnvironment">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="userName" type="xsd:string" />

  <xsd:element minOccurs="0" name="password" type="xsd:string" />

  <xsd:element minOccurs="0" name="environmentName" type="xsd:string" />

  <xsd:element minOccurs="0" name="environmentDescription" type="xsd:string" />

  <xsd:element minOccurs="0" name="applicationName" type="xsd:string" />

  <xsd:element minOccurs="0" name="architectureName" type="xsd:string" />

  </xsd:sequence>

  </xsd:complexType>

  <xsd:element name="createEnvironmentResponse"
type="tns:createEnvironmentResponse" />

- <xsd:complexType name="createEnvironmentResponse">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="return" nillable="true" type="xsd:long" />

  </xsd:sequence>

  </xsd:complexType>

  <xsd:element name="getProcessStatusAsString" type="tns:getProcessStatusAsString"
/>

- <xsd:complexType name="getProcessStatusAsString">
```

```
- <xsd:sequence>

  <xsd:element minOccurs="0" name="arg0" nillable="true" type="xsd:long" />

  </xsd:sequence>

  </xsd:complexType>

  <xsd:element name="getProcessStatusAsStringResponse"
type="tns:getProcessStatusAsStringResponse" />

- <xsd:complexType name="getProcessStatusAsStringResponse">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="return" nillable="true" type="xsd:string" />

  </xsd:sequence>

  </xsd:complexType>

  <xsd:element name="getAllApplications" type="tns:getAllApplications" />

- <xsd:complexType name="getAllApplications">

- <xsd:sequence>

  <xsd:element minOccurs="0" name="username" type="xsd:string" />

  <xsd:element minOccurs="0" name="password" type="xsd:string" />

  </xsd:sequence>

  </xsd:complexType>

  <xsd:element name="getAllApplicationsResponse"
type="tns:getAllApplicationsResponse" />

- <xsd:complexType name="getAllApplicationsResponse">

- <xsd:sequence>

  <xsd:element name="return" nillable="true" type="ns0:ArrayOfApplicationWS" />

  </xsd:sequence>

  </xsd:complexType>

  <xsd:element name="AuthenticationWSException" nillable="true"
type="ns1:AuthenticationWSException" />

  <xsd:element name="NolioWSException" nillable="true" type="ns1:NolioWSException"
/>

  <xsd:element name="ApplicationNotExistWSException" nillable="true"
type="ns1:ApplicationNotExistWSException" />
```

```
  <xsd:element name="EnvironmentNotExistWSException" nillable="true"
type="ns1:EnvironmentNotExistWSException" />

  <xsd:element name="RCNotificationException" nillable="true"
type="ns2:RCNotificationException" />

  <xsd:element name="JobNotExistWSException" nillable="true"
type="ns1:JobNotExistWSException" />

  <xsd:element name="ParameterAssignmentWSException" nillable="true"
type="ns1:ParameterAssignmentWSException" />

  <xsd:element name="AssignedProcessNotExistWSException" nillable="true"
type="ns1:AssignedProcessNotExistWSException" />

  <xsd:element name="AgentInstanceWSException" nillable="true"
type="ns1:AgentInstanceWSException" />

  <xsd:element name="DependenciesMissMatchWSException" nillable="true"
type="ns1:DependenciesMissMatchWSException" />

  <xsd:element name="ProcessRunFailedWSException" nillable="true"
type="ns1:ProcessRunFailedWSException" />

  <xsd:element name="JobCreationWSException" nillable="true"
type="ns1:JobCreationWSException" />

  </xsd:schema>

- <xsd:schema attributeFormDefault="qualified" elementFormDefault="qualified"
targetNamespace="http://notifications.rc.api.dataservices.server.platform.nolio.c
om"
xmlns:tns="http://notifications.rc.api.dataservices.server.platform.nolio.com"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">

- <xsd:complexType name="RCNotificationException">

  <xsd:sequence />

  </xsd:complexType>

  </xsd:schema>

- <xsd:schema attributeFormDefault="qualified" elementFormDefault="qualified"
targetNamespace="http://exceptions.webservice.model.api.dataservices.server.platf
orm.nolio.com"
xmlns:tns="http://exceptions.webservice.model.api.dataservices.server.platform.no
lio.com" xmlns:xsd="http://www.w3.org/2001/XMLSchema">

- <xsd:complexType name="ProcessRunFailedWSException">

  <xsd:sequence />

  </xsd:complexType>
```

```
- <xsd:complexType name="AgentInstanceWSException">
  <xsd:sequence />
  </xsd:complexType>
- <xsd:complexType name="JobCreationWSException">
  <xsd:sequence />
  </xsd:complexType>
- <xsd:complexType name="EnvironmentNotExistWSException">
  <xsd:sequence />
  </xsd:complexType>
- <xsd:complexType name="NolioWSException">
  <xsd:sequence />
  </xsd:complexType>
- <xsd:complexType name="ApplicationNotExistWSException">
  <xsd:sequence />
  </xsd:complexType>
- <xsd:complexType name="DependenciesMissMatchWSException">
  <xsd:sequence />
  </xsd:complexType>
- <xsd:complexType name="AuthenticationWSException">
  <xsd:sequence />
  </xsd:complexType>
- <xsd:complexType name="AssignedProcessNotExistWSException">
  <xsd:sequence />
  </xsd:complexType>
- <xsd:complexType name="ParameterAssignmentWSException">
  <xsd:sequence />
  </xsd:complexType>
- <xsd:complexType name="JobNotExistWSException">
```

```
  <xsd:sequence />

  </xsd:complexType>

  </xsd:schema>

  </wsdl:types>

- <wsdl:message name="getAllApplicationsResponse">

  <wsdl:part element="tns:getAllApplicationsResponse" name="parameters" />

  </wsdl:message>

- <wsdl:message name="DependenciesMissMatchWSException">

  <wsdl:part element="tns:DependenciesMissMatchWSException"
name="DependenciesMissMatchWSException" />

  </wsdl:message>

- <wsdl:message name="JobNotExistWSException">

  <wsdl:part element="tns:JobNotExistWSException" name="JobNotExistWSException" />

  </wsdl:message>

- <wsdl:message name="scheduleProcessRun">

  <wsdl:part element="tns:scheduleProcessRun" name="parameters" />

  </wsdl:message>

- <wsdl:message name="stopJobResponse">

  <wsdl:part element="tns:stopJobResponse" name="parameters" />

  </wsdl:message>

- <wsdl:message name="getAllAgentsResponse">

  <wsdl:part element="tns:getAllAgentsResponse" name="parameters" />

  </wsdl:message>

- <wsdl:message name="removeTagFromEnvironment">

  <wsdl:part element="tns:removeTagFromEnvironment" name="parameters" />

  </wsdl:message>

- <wsdl:message name="runProcess2Response">

  <wsdl:part element="tns:runProcess2Response" name="parameters" />

  </wsdl:message>
```

```
- <wsdl:message name="runProcess2">
  <wsdl:part element="tns:runProcess2" name="parameters" />
  </wsdl:message>
- <wsdl:message name="mapInstanceToEnvironmentResponse">
  <wsdl:part element="tns:mapInstanceToEnvironmentResponse" name="parameters" />
  </wsdl:message>
- <wsdl:message name="createEnvironment">
  <wsdl:part element="tns:createEnvironment" name="parameters" />
  </wsdl:message>
- <wsdl:message name="ApplicationNotExistWSException">
  <wsdl:part element="tns:ApplicationNotExistWSException"
name="ApplicationNotExistWSException" />
  </wsdl:message>
- <wsdl:message name="getUserInputParametersResponse">
  <wsdl:part element="tns:getUserInputParametersResponse" name="parameters" />
  </wsdl:message>
- <wsdl:message name="getEnvironmentsForApplicationResponse">
  <wsdl:part element="tns:getEnvironmentsForApplicationResponse" name="parameters"
/>
  </wsdl:message>
- <wsdl:message name="getAssignedProcessesForEnvironmentResponse">
  <wsdl:part element="tns:getAssignedProcessesForEnvironmentResponse"
name="parameters" />
  </wsdl:message>
- <wsdl:message name="runProcessResponse">
  <wsdl:part element="tns:runProcessResponse" name="parameters" />
  </wsdl:message>
- <wsdl:message name="getConnectedESForAgentResponse">
  <wsdl:part element="tns:getConnectedESForAgentResponse" name="parameters" />
  </wsdl:message>
```

```
- <wsdl:message name="RCNotificationException">
  <wsdl:part element="tns:RCNotificationException" name="RCNotificationException"
/>
  </wsdl:message>
- <wsdl:message name="scheduleProcessRunResponse">
  <wsdl:part element="tns:scheduleProcessRunResponse" name="parameters" />
  </wsdl:message>
- <wsdl:message name="getEnvironmentServersResponse">
  <wsdl:part element="tns:getEnvironmentServersResponse" name="parameters" />
  </wsdl:message>
- <wsdl:message name="getEnvironmentServers">
  <wsdl:part element="tns:getEnvironmentServers" name="parameters" />
  </wsdl:message>
- <wsdl:message name="stopJob">
  <wsdl:part element="tns:stopJob" name="parameters" />
  </wsdl:message>
- <wsdl:message name="removeAllAgentsFromEnvironmentResponse">
  <wsdl:part element="tns:removeAllAgentsFromEnvironmentResponse"
name="parameters" />
  </wsdl:message>
- <wsdl:message name="assignTagToEnvironment">
  <wsdl:part element="tns:assignTagToEnvironment" name="parameters" />
  </wsdl:message>
- <wsdl:message name="getEnvironmentsForApplication">
  <wsdl:part element="tns:getEnvironmentsForApplication" name="parameters" />
  </wsdl:message>
- <wsdl:message name="ParameterAssignmentWSException">
  <wsdl:part element="tns:ParameterAssignmentWSException"
name="ParameterAssignmentWSException" />
  </wsdl:message>
```

```
- <wsdl:message name="getJobErrors">
  <wsdl:part element="tns:getJobErrors" name="parameters" />
  </wsdl:message>
- <wsdl:message name="getConnectedAgentsForES">
  <wsdl:part element="tns:getConnectedAgentsForES" name="parameters" />
  </wsdl:message>
- <wsdl:message name="ProcessRunFailedWSException">
  <wsdl:part element="tns:ProcessRunFailedWSException"
name="ProcessRunFailedWSException" />
  </wsdl:message>
- <wsdl:message name="AuthenticationWSException">
  <wsdl:part element="tns:AuthenticationWSException"
name="AuthenticationWSException" />
  </wsdl:message>
- <wsdl:message name="getProcessStatusAsString">
  <wsdl:part element="tns:getProcessStatusAsString" name="parameters" />
  </wsdl:message>
- <wsdl:message name="getAssignedProcessesForEnvironment">
  <wsdl:part element="tns:getAssignedProcessesForEnvironment" name="parameters" />
  </wsdl:message>
- <wsdl:message name="assignTagToEnvironmentResponse">
  <wsdl:part element="tns:assignTagToEnvironmentResponse" name="parameters" />
  </wsdl:message>
- <wsdl:message name="EnvironmentNotExistWSException">
  <wsdl:part element="tns:EnvironmentNotExistWSException"
name="EnvironmentNotExistWSException" />
  </wsdl:message>
- <wsdl:message name="exportApplications">
  <wsdl:part element="tns:exportApplications" name="parameters" />
  </wsdl:message>
```

```
- <wsdl:message name="getConnectedESForAgent">
  <wsdl:part element="tns:getConnectedESForAgent" name="parameters" />
  </wsdl:message>
- <wsdl:message name="getJobFailedStepsResponse">
  <wsdl:part element="tns:getJobFailedStepsResponse" name="parameters" />
  </wsdl:message>
- <wsdl:message name="JobCreationWSException">
  <wsdl:part element="tns:JobCreationWSException" name="JobCreationWSException" />
  </wsdl:message>
- <wsdl:message name="getUserInputParameters">
  <wsdl:part element="tns:getUserInputParameters" name="parameters" />
  </wsdl:message>
- <wsdl:message name="NolioWSException">
  <wsdl:part element="tns:NolioWSException" name="NolioWSException" />
  </wsdl:message>
- <wsdl:message name="createEnvironmentResponse">
  <wsdl:part element="tns:createEnvironmentResponse" name="parameters" />
  </wsdl:message>
- <wsdl:message name="removeInstanceFromEnvironmentResponse">
  <wsdl:part element="tns:removeInstanceFromEnvironmentResponse" name="parameters"
/>
  </wsdl:message>
- <wsdl:message name="removeAllAgentsFromEnvironment">
  <wsdl:part element="tns:removeAllAgentsFromEnvironment" name="parameters" />
  </wsdl:message>
- <wsdl:message name="removeTagFromEnvironmentResponse">
  <wsdl:part element="tns:removeTagFromEnvironmentResponse" name="parameters" />
  </wsdl:message>
- <wsdl:message name="getAllApplications">
```

```
  <wsdl:part element="tns:getAllApplications" name="parameters" />

  </wsdl:message>

- <wsdl:message name="getJobStatusResponse">

  <wsdl:part element="tns:getJobStatusResponse" name="parameters" />

  </wsdl:message>

- <wsdl:message name="getJobFailedSteps">

  <wsdl:part element="tns:getJobFailedSteps" name="parameters" />

  </wsdl:message>

- <wsdl:message name="AssignedProcessNotExistWSException">

  <wsdl:part element="tns:AssignedProcessNotExistWSException"
name="AssignedProcessNotExistWSException" />

  </wsdl:message>

- <wsdl:message name="removeInstanceFromEnvironment">

  <wsdl:part element="tns:removeInstanceFromEnvironment" name="parameters" />

  </wsdl:message>

- <wsdl:message name="getJobStatus">

  <wsdl:part element="tns:getJobStatus" name="parameters" />

  </wsdl:message>

- <wsdl:message name="runProcess">

  <wsdl:part element="tns:runProcess" name="parameters" />

  </wsdl:message>

- <wsdl:message name="AgentInstanceWSException">

  <wsdl:part element="tns:AgentInstanceWSException"
name="AgentInstanceWSException" />

  </wsdl:message>

- <wsdl:message name="exportApplicationsResponse">

  <wsdl:part element="tns:exportApplicationsResponse" name="parameters" />

  </wsdl:message>

- <wsdl:message name="mapInstanceToEnvironment">
```

```
  <wsdl:part element="tns:mapInstanceToEnvironment" name="parameters" />
  </wsdl:message>
- <wsdl:message name="getJobErrorsResponse">
  <wsdl:part element="tns:getJobErrorsResponse" name="parameters" />
  </wsdl:message>
- <wsdl:message name="getProcessStatusAsStringResponse">
  <wsdl:part element="tns:getProcessStatusAsStringResponse" name="parameters" />
  </wsdl:message>
- <wsdl:message name="getAllAgents">
  <wsdl:part element="tns:getAllAgents" name="parameters" />
  </wsdl:message>
- <wsdl:message name="getConnectedAgentsForESResponse">
  <wsdl:part element="tns:getConnectedAgentsForESResponse" name="parameters" />
  </wsdl:message>
- <wsdl:portType name="OpenAPIServicePortType">
- <wsdl:operation name="removeInstanceFromEnvironment">
  <wsdl:input message="tns:removeInstanceFromEnvironment"
name="removeInstanceFromEnvironment" />
  <wsdl:output message="tns:removeInstanceFromEnvironmentResponse"
name="removeInstanceFromEnvironmentResponse" />
  </wsdl:operation>
- <wsdl:operation name="getAllAgents">
  <wsdl:input message="tns:getAllAgents" name="getAllAgents" />
  <wsdl:output message="tns:getAllAgentsResponse" name="getAllAgentsResponse" />
  <wsdl:fault message="tns:NolioWSException" name="NolioWSException" />
  <wsdl:fault message="tns:AuthenticationWSException"
name="AuthenticationWSException" />
  </wsdl:operation>
- <wsdl:operation name="getConnectedAgentsForES">
```

```
  <wsdl:input message="tns:getConnectedAgentsForES" name="getConnectedAgentsForES"
/>

  <wsdl:output message="tns:getConnectedAgentsForESResponse"
name="getConnectedAgentsForESResponse" />

  <wsdl:fault message="tns:NolioWSException" name="NolioWSException" />

  <wsdl:fault message="tns:AuthenticationWSException"
name="AuthenticationWSException" />

  </wsdl:operation>

- <wsdl:operation name="assignTagToEnvironment">

  <wsdl:input message="tns:assignTagToEnvironment" name="assignTagToEnvironment"
/>

  <wsdl:output message="tns:assignTagToEnvironmentResponse"
name="assignTagToEnvironmentResponse" />

  <wsdl:fault message="tns:EnvironmentNotExistWSException"
name="EnvironmentNotExistWSException" />

  <wsdl:fault message="tns:RCNotificationException" name="RCNotificationException"
/>

  <wsdl:fault message="tns:ApplicationNotExistWSException"
name="ApplicationNotExistWSException" />

  </wsdl:operation>

- <wsdl:operation name="getJobFailedSteps">

  <wsdl:input message="tns:getJobFailedSteps" name="getJobFailedSteps" />

  <wsdl:output message="tns:getJobFailedStepsResponse"
name="getJobFailedStepsResponse" />

  <wsdl:fault message="tns:NolioWSException" name="NolioWSException" />

  <wsdl:fault message="tns:JobNotExistWSException" name="JobNotExistWSException"
/>

  <wsdl:fault message="tns:AuthenticationWSException"
name="AuthenticationWSException" />

  </wsdl:operation>

- <wsdl:operation name="getConnectedESForAgent">

  <wsdl:input message="tns:getConnectedESForAgent" name="getConnectedESForAgent"
/>
```

```
  <wsdl:output message="tns:getConnectedESForAgentResponse"
name="getConnectedESForAgentResponse" />

  <wsdl:fault message="tns:NolioWSException" name="NolioWSException" />

  <wsdl:fault message="tns:AuthenticationWSException"
name="AuthenticationWSException" />

  </wsdl:operation>

- <wsdl:operation name="removeAllAgentsFromEnvironment">

  <wsdl:input message="tns:removeAllAgentsFromEnvironment"
name="removeAllAgentsFromEnvironment" />

  <wsdl:output message="tns:removeAllAgentsFromEnvironmentResponse"
name="removeAllAgentsFromEnvironmentResponse" />

  </wsdl:operation>

- <wsdl:operation name="getJobErrors">

  <wsdl:input message="tns:getJobErrors" name="getJobErrors" />

  <wsdl:output message="tns:getJobErrorsResponse" name="getJobErrorsResponse" />

  <wsdl:fault message="tns:NolioWSException" name="NolioWSException" />

  <wsdl:fault message="tns:JobNotExistWSException" name="JobNotExistWSException"
/>

  <wsdl:fault message="tns:AuthenticationWSException"
name="AuthenticationWSException" />

  </wsdl:operation>

- <wsdl:operation name="exportApplications">

  <wsdl:input message="tns:exportApplications" name="exportApplications" />

  <wsdl:output message="tns:exportApplicationsResponse"
name="exportApplicationsResponse" />

  <wsdl:fault message="tns:NolioWSException" name="NolioWSException" />

  </wsdl:operation>

- <wsdl:operation name="getJobStatus">

  <wsdl:input message="tns:getJobStatus" name="getJobStatus" />

  <wsdl:output message="tns:getJobStatusResponse" name="getJobStatusResponse" />

  <wsdl:fault message="tns:NolioWSException" name="NolioWSException" />
```

```
  <wsdl:fault message="tns:JobNotExistWSException" name="JobNotExistWSException"
/>

  <wsdl:fault message="tns:AuthenticationWSException"
name="AuthenticationWSException" />

  </wsdl:operation>

- <wsdl:operation name="scheduleProcessRun">

  <wsdl:input message="tns:scheduleProcessRun" name="scheduleProcessRun" />

  <wsdl:output message="tns:scheduleProcessRunResponse"
name="scheduleProcessRunResponse" />

  <wsdl:fault message="tns:ParameterAssignmentWSException"
name="ParameterAssignmentWSException" />

  <wsdl:fault message="tns:NolioWSException" name="NolioWSException" />

  <wsdl:fault message="tns:AssignedProcessNotExistWSException"
name="AssignedProcessNotExistWSException" />

  <wsdl:fault message="tns:EnvironmentNotExistWSException"
name="EnvironmentNotExistWSException" />

  <wsdl:fault message="tns:AgentInstanceWSException"
name="AgentInstanceWSException" />

  <wsdl:fault message="tns:AuthenticationWSException"
name="AuthenticationWSException" />

  <wsdl:fault message="tns:ApplicationNotExistWSException"
name="ApplicationNotExistWSException" />

  </wsdl:operation>

- <wsdl:operation name="getAssignedProcessesForEnvironment">

  <wsdl:input message="tns:getAssignedProcessesForEnvironment"
name="getAssignedProcessesForEnvironment" />

  <wsdl:output message="tns:getAssignedProcessesForEnvironmentResponse"
name="getAssignedProcessesForEnvironmentResponse" />

  <wsdl:fault message="tns:NolioWSException" name="NolioWSException" />

  <wsdl:fault message="tns:EnvironmentNotExistWSException"
name="EnvironmentNotExistWSException" />

  <wsdl:fault message="tns:AuthenticationWSException"
name="AuthenticationWSException" />
```

```
  <wsdl:fault message="tns:ApplicationNotExistWSException"
name="ApplicationNotExistWSException" />

  </wsdl:operation>

- <wsdl:operation name="removeTagFromEnvironment">

  <wsdl:input message="tns:removeTagFromEnvironment"
name="removeTagFromEnvironment" />

  <wsdl:output message="tns:removeTagFromEnvironmentResponse"
name="removeTagFromEnvironmentResponse" />

  <wsdl:fault message="tns:EnvironmentNotExistWSException"
name="EnvironmentNotExistWSException" />

  <wsdl:fault message="tns:RCNotificationException" name="RCNotificationException"
/>

  <wsdl:fault message="tns:ApplicationNotExistWSException"
name="ApplicationNotExistWSException" />

  </wsdl:operation>

- <wsdl:operation name="getEnvironmentServers">

  <wsdl:input message="tns:getEnvironmentServers" name="getEnvironmentServers" />

  <wsdl:output message="tns:getEnvironmentServersResponse"
name="getEnvironmentServersResponse" />

  <wsdl:fault message="tns:NolioWSException" name="NolioWSException" />

  <wsdl:fault message="tns:EnvironmentNotExistWSException"
name="EnvironmentNotExistWSException" />

  <wsdl:fault message="tns:AuthenticationWSException"
name="AuthenticationWSException" />

  <wsdl:fault message="tns:ApplicationNotExistWSException"
name="ApplicationNotExistWSException" />

  </wsdl:operation>

- <wsdl:operation name="getEnvironmentsForApplication">

  <wsdl:input message="tns:getEnvironmentsForApplication"
name="getEnvironmentsForApplication" />

  <wsdl:output message="tns:getEnvironmentsForApplicationResponse"
name="getEnvironmentsForApplicationResponse" />

  <wsdl:fault message="tns:NolioWSException" name="NolioWSException" />
```

```
  <wsdl:fault message="tns:AuthenticationWSException"
name="AuthenticationWSException" />

  <wsdl:fault message="tns:ApplicationNotExistWSException"
name="ApplicationNotExistWSException" />

  </wsdl:operation>

- <wsdl:operation name="runProcess2">

  <wsdl:input message="tns:runProcess2" name="runProcess2" />

  <wsdl:output message="tns:runProcess2Response" name="runProcess2Response" />

  <wsdl:fault message="tns:NolioWSException" name="NolioWSException" />

  <wsdl:fault message="tns:DependenciesMissMatchWSException"
name="DependenciesMissMatchWSException" />

  <wsdl:fault message="tns:AuthenticationWSException"
name="AuthenticationWSException" />

  <wsdl:fault message="tns:ProcessRunFailedWSException"
name="ProcessRunFailedWSException" />

  <wsdl:fault message="tns:JobCreationWSException" name="JobCreationWSException"
/>

  </wsdl:operation>

- <wsdl:operation name="runProcess">

  <wsdl:input message="tns:runProcess" name="runProcess" />

  <wsdl:output message="tns:runProcessResponse" name="runProcessResponse" />

  </wsdl:operation>

- <wsdl:operation name="getUserInputParameters">

  <wsdl:input message="tns:getUserInputParameters" name="getUserInputParameters"
/>

  <wsdl:output message="tns:getUserInputParametersResponse"
name="getUserInputParametersResponse" />

  <wsdl:fault message="tns:NolioWSException" name="NolioWSException" />

  <wsdl:fault message="tns:AssignedProcessNotExistWSException"
name="AssignedProcessNotExistWSException" />

  <wsdl:fault message="tns:EnvironmentNotExistWSException"
name="EnvironmentNotExistWSException" />
```

```
  <wsdl:fault message="tns:AuthenticationWSException"
name="AuthenticationWSException" />

  <wsdl:fault message="tns:ApplicationNotExistWSException"
name="ApplicationNotExistWSException" />

  </wsdl:operation>

- <wsdl:operation name="stopJob">

  <wsdl:input message="tns:stopJob" name="stopJob" />

  <wsdl:output message="tns:stopJobResponse" name="stopJobResponse" />

  <wsdl:fault message="tns:NolioWSException" name="NolioWSException" />

  <wsdl:fault message="tns:JobNotExistWSException" name="JobNotExistWSException"
/>

  <wsdl:fault message="tns:AuthenticationWSException"
name="AuthenticationWSException" />

  </wsdl:operation>

- <wsdl:operation name="mapInstanceToEnvironment">

  <wsdl:input message="tns:mapInstanceToEnvironment"
name="mapInstanceToEnvironment" />

  <wsdl:output message="tns:mapInstanceToEnvironmentResponse"
name="mapInstanceToEnvironmentResponse" />

  </wsdl:operation>

- <wsdl:operation name="createEnvironment">

  <wsdl:input message="tns:createEnvironment" name="createEnvironment" />

  <wsdl:output message="tns:createEnvironmentResponse"
name="createEnvironmentResponse" />

  </wsdl:operation>

- <wsdl:operation name="getProcessStatusAsString">

  <wsdl:input message="tns:getProcessStatusAsString"
name="getProcessStatusAsString" />

  <wsdl:output message="tns:getProcessStatusAsStringResponse"
name="getProcessStatusAsStringResponse" />

  </wsdl:operation>

- <wsdl:operation name="getAllApplications">
```

```
  <wsdl:input message="tns:getAllApplications" name="getAllApplications" />

  <wsdl:output message="tns:getAllApplicationsResponse"
name="getAllApplicationsResponse" />

  <wsdl:fault message="tns:NolioWSException" name="NolioWSException" />

  <wsdl:fault message="tns:AuthenticationWSException"
name="AuthenticationWSException" />

  </wsdl:operation>

  </wsdl:portType>

- <wsdl:binding name="OpenAPIServiceSoapBinding"
type="tns:OpenAPIServicePortType">

  <soap12:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http" />

- <wsdl:operation name="removeInstanceFromEnvironment">

  <soap12:operation soapAction="" style="document" />

- <wsdl:input name="removeInstanceFromEnvironment">

  <soap12:body use="literal" />

  </wsdl:input>

- <wsdl:output name="removeInstanceFromEnvironmentResponse">

  <soap12:body use="literal" />

  </wsdl:output>

  </wsdl:operation>

- <wsdl:operation name="getAllAgents">

  <soap12:operation soapAction="" style="document" />

- <wsdl:input name="getAllAgents">

  <soap12:body use="literal" />

  </wsdl:input>

- <wsdl:output name="getAllAgentsResponse">

  <soap12:body use="literal" />

  </wsdl:output>

- <wsdl:fault name="NolioWSException">
```

```
  <soap12:fault name="NolioWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="AuthenticationWSException">
  <soap12:fault name="AuthenticationWSException" use="literal" />
  </wsdl:fault>
  </wsdl:operation>
- <wsdl:operation name="getConnectedAgentsForES">
  <soap12:operation soapAction="" style="document" />
- <wsdl:input name="getConnectedAgentsForES">
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output name="getConnectedAgentsForESResponse">
  <soap12:body use="literal" />
  </wsdl:output>
- <wsdl:fault name="NolioWSException">
  <soap12:fault name="NolioWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="AuthenticationWSException">
  <soap12:fault name="AuthenticationWSException" use="literal" />
  </wsdl:fault>
  </wsdl:operation>
- <wsdl:operation name="assignTagToEnvironment">
  <soap12:operation soapAction="" style="document" />
- <wsdl:input name="assignTagToEnvironment">
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output name="assignTagToEnvironmentResponse">
  <soap12:body use="literal" />
```

```
  </wsdl:output>
- <wsdl:fault name="EnvironmentNotExistWSException">
  <soap12:fault name="EnvironmentNotExistWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="RCNotificationException">
  <soap12:fault name="RCNotificationException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="ApplicationNotExistWSException">
  <soap12:fault name="ApplicationNotExistWSException" use="literal" />
  </wsdl:fault>
  </wsdl:operation>
- <wsdl:operation name="getConnectedESForAgent">
  <soap12:operation soapAction="" style="document" />
- <wsdl:input name="getConnectedESForAgent">
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output name="getConnectedESForAgentResponse">
  <soap12:body use="literal" />
  </wsdl:output>
- <wsdl:fault name="NolioWSException">
  <soap12:fault name="NolioWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="AuthenticationWSException">
  <soap12:fault name="AuthenticationWSException" use="literal" />
  </wsdl:fault>
  </wsdl:operation>
- <wsdl:operation name="getJobFailedSteps">
  <soap12:operation soapAction="" style="document" />
```

```
- <wsdl:input name="getJobFailedSteps">
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output name="getJobFailedStepsResponse">
  <soap12:body use="literal" />
  </wsdl:output>
- <wsdl:fault name="NolioWSException">
  <soap12:fault name="NolioWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="JobNotExistWSException">
  <soap12:fault name="JobNotExistWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="AuthenticationWSException">
  <soap12:fault name="AuthenticationWSException" use="literal" />
  </wsdl:fault>
  </wsdl:operation>
- <wsdl:operation name="removeAllAgentsFromEnvironment">
  <soap12:operation soapAction="" style="document" />
- <wsdl:input name="removeAllAgentsFromEnvironment">
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output name="removeAllAgentsFromEnvironmentResponse">
  <soap12:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="getJobErrors">
  <soap12:operation soapAction="" style="document" />
- <wsdl:input name="getJobErrors">
```

```
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output name="getJobErrorsResponse">
  <soap12:body use="literal" />
  </wsdl:output>
- <wsdl:fault name="NolioWSException">
  <soap12:fault name="NolioWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="JobNotExistWSException">
  <soap12:fault name="JobNotExistWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="AuthenticationWSException">
  <soap12:fault name="AuthenticationWSException" use="literal" />
  </wsdl:fault>
  </wsdl:operation>
- <wsdl:operation name="exportApplications">
  <soap12:operation soapAction="" style="document" />
- <wsdl:input name="exportApplications">
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output name="exportApplicationsResponse">
  <soap12:body use="literal" />
  </wsdl:output>
- <wsdl:fault name="NolioWSException">
  <soap12:fault name="NolioWSException" use="literal" />
  </wsdl:fault>
  </wsdl:operation>
- <wsdl:operation name="getJobStatus">
```

```
  <soap12:operation soapAction="" style="document" />
- <wsdl:input name="getJobStatus">
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output name="getJobStatusResponse">
  <soap12:body use="literal" />
  </wsdl:output>
- <wsdl:fault name="NolioWSException">
  <soap12:fault name="NolioWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="JobNotExistWSException">
  <soap12:fault name="JobNotExistWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="AuthenticationWSException">
  <soap12:fault name="AuthenticationWSException" use="literal" />
  </wsdl:fault>
  </wsdl:operation>
- <wsdl:operation name="scheduleProcessRun">
  <soap12:operation soapAction="" style="document" />
- <wsdl:input name="scheduleProcessRun">
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output name="scheduleProcessRunResponse">
  <soap12:body use="literal" />
  </wsdl:output>
- <wsdl:fault name="ParameterAssignmentWSException">
  <soap12:fault name="ParameterAssignmentWSException" use="literal" />
  </wsdl:fault>
```

```
- <wsdl:fault name="NolioWSException">
  <soap12:fault name="NolioWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="AssignedProcessNotExistWSException">
  <soap12:fault name="AssignedProcessNotExistWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="EnvironmentNotExistWSException">
  <soap12:fault name="EnvironmentNotExistWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="AgentInstanceWSException">
  <soap12:fault name="AgentInstanceWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="AuthenticationWSException">
  <soap12:fault name="AuthenticationWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="ApplicationNotExistWSException">
  <soap12:fault name="ApplicationNotExistWSException" use="literal" />
  </wsdl:fault>
  </wsdl:operation>
- <wsdl:operation name="getAssignedProcessesForEnvironment">
  <soap12:operation soapAction="" style="document" />
- <wsdl:input name="getAssignedProcessesForEnvironment">
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output name="getAssignedProcessesForEnvironmentResponse">
  <soap12:body use="literal" />
  </wsdl:output>
- <wsdl:fault name="NolioWSException">
```

```
  <soap12:fault name="NolioWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="EnvironmentNotExistWSException">
  <soap12:fault name="EnvironmentNotExistWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="AuthenticationWSException">
  <soap12:fault name="AuthenticationWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="ApplicationNotExistWSException">
  <soap12:fault name="ApplicationNotExistWSException" use="literal" />
  </wsdl:fault>
  </wsdl:operation>
- <wsdl:operation name="removeTagFromEnvironment">
  <soap12:operation soapAction="" style="document" />
- <wsdl:input name="removeTagFromEnvironment">
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output name="removeTagFromEnvironmentResponse">
  <soap12:body use="literal" />
  </wsdl:output>
- <wsdl:fault name="EnvironmentNotExistWSException">
  <soap12:fault name="EnvironmentNotExistWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="RCNotificationException">
  <soap12:fault name="RCNotificationException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="ApplicationNotExistWSException">
  <soap12:fault name="ApplicationNotExistWSException" use="literal" />
```

```
  </wsdl:fault>
  </wsdl:operation>
- <wsdl:operation name="getEnvironmentServers">
  <soap12:operation soapAction="" style="document" />
- <wsdl:input name="getEnvironmentServers">
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output name="getEnvironmentServersResponse">
  <soap12:body use="literal" />
  </wsdl:output>
- <wsdl:fault name="NolioWSException">
  <soap12:fault name="NolioWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="EnvironmentNotExistWSException">
  <soap12:fault name="EnvironmentNotExistWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="AuthenticationWSException">
  <soap12:fault name="AuthenticationWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="ApplicationNotExistWSException">
  <soap12:fault name="ApplicationNotExistWSException" use="literal" />
  </wsdl:fault>
  </wsdl:operation>
- <wsdl:operation name="getEnvironmentsForApplication">
  <soap12:operation soapAction="" style="document" />
- <wsdl:input name="getEnvironmentsForApplication">
  <soap12:body use="literal" />
  </wsdl:input>
```

```
- <wsdl:output name="getEnvironmentsForApplicationResponse">
  <soap12:body use="literal" />
  </wsdl:output>
- <wsdl:fault name="NolioWSException">
  <soap12:fault name="NolioWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="AuthenticationWSException">
  <soap12:fault name="AuthenticationWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="ApplicationNotExistWSException">
  <soap12:fault name="ApplicationNotExistWSException" use="literal" />
  </wsdl:fault>
  </wsdl:operation>
- <wsdl:operation name="runProcess">
  <soap12:operation soapAction="" style="document" />
- <wsdl:input name="runProcess">
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output name="runProcessResponse">
  <soap12:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="runProcess2">
  <soap12:operation soapAction="" style="document" />
- <wsdl:input name="runProcess2">
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output name="runProcess2Response">
```

```
  <soap12:body use="literal" />
  </wsdl:output>
- <wsdl:fault name="NolioWSException">
  <soap12:fault name="NolioWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="DependenciesMissMatchWSException">
  <soap12:fault name="DependenciesMissMatchWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="AuthenticationWSException">
  <soap12:fault name="AuthenticationWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="ProcessRunFailedWSException">
  <soap12:fault name="ProcessRunFailedWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="JobCreationWSException">
  <soap12:fault name="JobCreationWSException" use="literal" />
  </wsdl:fault>
  </wsdl:operation>
- <wsdl:operation name="getUserInputParameters">
  <soap12:operation soapAction="" style="document" />
- <wsdl:input name="getUserInputParameters">
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output name="getUserInputParametersResponse">
  <soap12:body use="literal" />
  </wsdl:output>
- <wsdl:fault name="NolioWSException">
  <soap12:fault name="NolioWSException" use="literal" />
```

```
  </wsdl:fault>
- <wsdl:fault name="AssignedProcessNotExistWSException">
  <soap12:fault name="AssignedProcessNotExistWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="EnvironmentNotExistWSException">
  <soap12:fault name="EnvironmentNotExistWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="AuthenticationWSException">
  <soap12:fault name="AuthenticationWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="ApplicationNotExistWSException">
  <soap12:fault name="ApplicationNotExistWSException" use="literal" />
  </wsdl:fault>
  </wsdl:operation>
- <wsdl:operation name="stopJob">
  <soap12:operation soapAction="" style="document" />
- <wsdl:input name="stopJob">
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output name="stopJobResponse">
  <soap12:body use="literal" />
  </wsdl:output>
- <wsdl:fault name="NolioWSException">
  <soap12:fault name="NolioWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="JobNotExistWSException">
  <soap12:fault name="JobNotExistWSException" use="literal" />
  </wsdl:fault>
```

```
- <wsdl:fault name="AuthenticationWSException">
  <soap12:fault name="AuthenticationWSException" use="literal" />
  </wsdl:fault>
  </wsdl:operation>
- <wsdl:operation name="createEnvironment">
  <soap12:operation soapAction="" style="document" />
- <wsdl:input name="createEnvironment">
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output name="createEnvironmentResponse">
  <soap12:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="mapInstanceToEnvironment">
  <soap12:operation soapAction="" style="document" />
- <wsdl:input name="mapInstanceToEnvironment">
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output name="mapInstanceToEnvironmentResponse">
  <soap12:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="getProcessStatusAsString">
  <soap12:operation soapAction="" style="document" />
- <wsdl:input name="getProcessStatusAsString">
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output name="getProcessStatusAsStringResponse">
```

```
  <soap12:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
- <wsdl:operation name="getAllApplications">
  <soap12:operation soapAction="" style="document" />
- <wsdl:input name="getAllApplications">
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output name="getAllApplicationsResponse">
  <soap12:body use="literal" />
  </wsdl:output>
- <wsdl:fault name="NolioWSException">
  <soap12:fault name="NolioWSException" use="literal" />
  </wsdl:fault>
- <wsdl:fault name="AuthenticationWSException">
  <soap12:fault name="AuthenticationWSException" use="literal" />
  </wsdl:fault>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:service name="OpenAPIService">
- <wsdl:port binding="tns:OpenAPIServiceSoapBinding" name="OpenAPIServicePort">
  <soap12:address
location="http://tamme012233:8080/datamanagement/ws/OpenAPIService" />
  </wsdl:port>
  </wsdl:service>
  </wsdl:definitions>
```

SOAP API Return Codes

-1 - GENERAL_ERROR_OCCURRED

-2 - ERROR_AUTHENTICATION_FAILED

-3 - ERROR_REMOTE_EXCEPTION_OCCURRED

-4 - ERROR_UNMAPPED_INSTANCE

-5 - ERROR_MISSING_SERVER

-6 - ERROR_BAD_PROC_NAME

-7 - ERROR_EXECUTION_FAILED

-8 - ERROR_DEPENDENCY_CONSTRAINT

-9 - ERROR_UPDATE_PARAMETER

-10 - ERROR_MISSING_PARAMETER_VALUE

-11 - ERROR_BAD_PARAMETER

-12 - ERROR_BAD_APP_NAME

-13 - ERROR_BAD_ENV_NAME

-14 - ERROR_ABNORMAL_TERMINATION

-15 - ERROR_MANUAL_OPERATION

-16 - ERROR_EXECUTION_TIMED_OUT

-17 - ERROR_AUTHORIZATION_FAILED

-18 - ERROR_GET_STATUS_FAILED

-19 - ERROR_CREATION_FAILED

-20 - ERROR_DUPLICATE_PROCESSES

-21 - ERROR_ENV_CONFIGURATION

0 - CREATING

1 - CREATION

2 - CREATION_FAILED

3 - CREATION_FINISHED

4 - BLOCKED

5 - INIT

6 - FILES_PROPAGATION_STARTING

7 - FILES_PROPAGATION_IN_PROGRESS

8 - FILES_PROPAGATION_FINISHED

9 - FILES_PROPAGATION_FAILED

10 - CONNECTIVITY_CHECK

11 - FILES_DISTRIBUTION

12 - FILES_DISTRIBUTION_FINISHED

13 - FILES_DISTRIBUTION_FAILED

14 - PRE_STARTING

15 - PRE_IN_PROGRESS

16 - PRE_PAUSING

17 - PRE_PAUSED

18 – PRE_FAILED_PAUSED

19 - PRE_STOPPING

20 - PRE_STOPPED

21 - PRE_FINISHED

22 - PRE_FAILED

23 - FLOW_IN_PROGRESS

24 - FLOW_PAUSING

25 - FLOW_PAUSED

26 - FLOW_FAILED_PAUSED

27 - FLOW_STOPPING

28 - FLOW_STOPPED

29 - FLOW_FINISHED

30 - RESUMING

Chapter 4: REST API Reference

CA Release Automation offers REST API enabling external systems to configure, execute and monitor release operations and accompanied artifacts without having to access the Release Operations Center UI. The REST API provides both the Get and the POST methods required to retrieve data and to execute process.

The base URL for the available services is:

http://<host>:< port>/datamanagement/a/api

The REST API documentation is available directly in your deployment of CA Release Automation at the following URL:

http://<host>:< port>/datamanagement/restApi.jsp

Note: For versions 4.7.0 and higher, any command containing /release*, add /v2 after the base URL. The values available with /v2 are specific to status and stage.

Note: REST calls should include Content-Type: text/html in the HTTP header.

Method	URL	Description
POST	/add-artifact-package-to-deployment-plan (see page 101)	Adds an artifact package to an existing deployment plan
GET	/applications (see page 102)	Retrieves all applications
GET	/applications/{appId} (see page 102)	Retrieves a specific application
GET	/applications/{appId}/environments (see page 104)	Retrieves all environments for a specific application
GET	/applications/{appId}/environments/{envId} (see page 104)	Retrieves a specific environment for a specific application
GET	/applications/{appId}/environments/{envId}/releases (see page 104)	Retrieves all deployments for a specific application and environment
GET	/applications/{appId}/environments/{envId}/releases/{releaseId} (see page 105)	Retrieves a specific deployment for a specific application and environment
GET	/applications/{appId}/projects (see page 106)	Retrieves all active projects for a specific application.
GET	/applications/{appId}projects/{projectId} (see page 107)	Retrieves a project from application and project ids.
GET	/applications/{appId}/projects/{projectId}/deployment-plans (see page 108)	Retrieves all deployment plans under specific application and project

Method	URL	Description
GET	/applications/{appId}/projects/{projectId}/deployments-plans/deploymentPlanId (see page 109)	Retrieves specific deployment plan
GET	/applications/{appId}/templates (see page 110)	Retrieves all templates for a specific application
GET	/applications/{appId}/templates/{templateId} (see page 111)	Retrieves a specific template for a specific application
POST	/artifact-details (see page 112)	Retrieves the details of a specific artifact **(V2)**
POST	/artifact-status (see page 113)	Retrieves the status of a specific artifact **(V2)**
POST	/artifact-version-details (see page 114)	Retrieves the details of a specific artifact-version **(V3+)**
POST	/artifact-version-status (see page 115)	Retrieves the status of a specific artifact-version **(V3+)**
POSt	/assign-new-artifact-package-to-deployment-plan (see page 116)	Creates and Adds an artifact package to an existing deployment plan
POST	/create-artifact (see page 117)	Creates an artifact **(V2)**
POST	/create-artifact-package (see page 118)	Creates an empty artifact package **(V3+)**
POST	/create-artifact-package-xml (see page 118)	Creates a new artifact package provided in XML
POST	/create-artifact-version (see page 119)	Creates an artifact-version with all the different parameters entered **(V3+)**
POST	/create-deployment-plan (see page 120)	Creates a deployment plan
POST	/create-project (see page 121)	Create a unique project for the given application.
POST	/create-release (see page 122)	Creating a release from an existing template
POST	/delete-release (see page 123)	Deleting a release
POST	/export-release (see page 124)	Exports a release
POST	/export-template (see page 126)	Exports a release to an XML
GET	/export/processes (see page 127)	Exports a process to a PDF
GET	/export/releases (see page 128)	Exports a release to a PDF
POST	/get-artifact-package-content (see page 129)	Retrieves artifact-versions for a given artifact-package **(V3+)**
POST	/get-artifact-versions (see page 130)	Retrieves artifact-versions for a given artifact-definition **(V3+)**
POST	/get-environment-parameter (see page 131)	Retrieves the value of a parameter in an environment
POST	/load-manifest (see page 132)	Loads a manifest file to a deployment plan

Method	URL	Description
POST	/release-status (see page 133)	Retrieves a status of a release
GET	/release-status/{releaseId} (see page 133)	Retrieves a status of a release
GET	/releases-reports (see page 135)	Retrieves the releases reports by filters
POST	/run-deployment-plan (see page 136)	Creates a deployment plan then runs deployments on different environments.
POST	/run-deployments (see page 137)	Creates (and optionally runs) a deployment
POST	/run-release (see page 139)	Run a release by entering a release id or unique release properties
POST	/run-template (see page 140)	Creates a release from a template, updates it and runs it.
POST	/schedule-release (see page 142)	Schedule a release by entering a release id or unique release attributes and the scheduled properties
POST	/step-status (see page 143)	Retrieves a step status
GET	/step-status/{stepId} (see page 143)	Retrieves a status of a step
POST	/stop-release (see page 143)	Stopping a running release
POST	/update-environment-parameter (see page 145)	Updates the value of a parameter in an environment
POST	/update-release (see page 146)	Update a release using XML

The REST API uses the following DTOs to standardize request/response data in REST calls:

- ApplicationApiDto (see page 147)
- ArtifactsApiDto (see page 147)
- ArtifactBasicApiDto (see page 148)
- ArtifactNameDtoList (see page 149)
- ArtifactPackageApiDto (see page 149)
- ArtifactPackageUploadDto (see page 149)
- ArtifactStatusApiDto (see page 150)
- ArtifactversionApiDto (see page 150)
- ArtifactVersionsApiDto (see page 150)
- CreateArtifactApiDto (see page 151)
- CreateArtifactFroDeploymentPlanDto (see page 152)
- CreateArtifactPackageApiDto (see page 152)
- CreateArtifactVersionApiDto (see page 153)
- CreateReleaseApiDto (see page 154)
- DeploymentApiDto (see page 155)
- DeploymentResponseApiDto (see page 156)
- DeploymentPlanApiDto (see page 156)
- DeploymentPlanResponseApiDto (see page 157)
- EnvironmentApiDto (see page 158)
- EnvironmentParameterApiDto (see page 159)
- FullEnvironmentParameterApiDto (see page 159)
- LoadManifestApiDto (see page 160)
- ProjectApiDto (see page 160)
- ReleaseApiDto (see page 161)
- ReleaseBasicApiDto (see page 162)
- ReleaseStatusApiDto (see page 162)
- ResponseData (see page 163)
- ResponseDataApiDto (see page 163)
- ReponseEnvironmentParameterApiDto (see page 163)
- RunReleaseApiDto (see page 164)
- RunTemplateApiDto (see page 165)

- ScheduleReleaseApiDto (see page 166)
- StepApiDto (see page 166)
- TemplateApiDto (see page 166)
- TemplateBasicApiDto (see page 167)
- UpdateReleaseApiDto (see page 168)

/add-artifact-package-to-deployment-plan

Adds an artifact package to an existing deployment plan. The required parameters are the deployment plan, project, build and application names with the artifact package name..

Resource URL

Type	Url and Format
POST	http://<host>:< port>datamanagement/a/api/<version>/add-artifact-package-to-deployment-plan

Request Parameter

Type	Description
DeploymentPlanApiDto (see page 156)	This dto contains the fields used for creating a deployment plan.

Response Parameters

Type	Description
DeploymentPlanResponseApiDto (see page 157)	This dto contains all the available values of a deployment plan.

Example

```
Command: /add-artifact-package-to-deployment-plan

Body:
{"deploymentPlan":"newDP","build":"1.0","project":"proj","application":"app","artifactPackage":"art pack"}

Response:

{"build": "1.0", "result": true, "project": "proj", "deploymentPlan": "newDP", "application": "app", "artifactPackage": "art pack", "deploymentTemplate": "newDT", "deploymentPlanId": "1", "templateCategory": "newTmp"}
```

/applications

Retrieves all applications.

Resource URL

Type	Url and Format
GET	http://<host>:< port>/datamanagement/a/api/<version>/applications

Response Parameters

Type	Description
ApplicationApiDto (see page 147)	The application dto that contains the application id, application name and description.

Example

```
Command: /applications

Response:

{"name":"Application
1","id":"2","description":""},{"name":"Myappname-old","id":"3","description":"(ex
ported)"},{"name":"Myappname","id":"4","description":"(exported)"}
```

/applications/{appId}

Retrieves a specific application by entering the application id in the path.

Resource URL

Type	Description
GET	http://<host>:< port>/datamanagement/a/api/<version>/applications/{appId}

Method Parameters

Parameters	Type	Description
appId	Long	Application ID

Response Parameters

Type	Description
ApplicationApiDto (see page 147)	The application dto that contains the application id, application name and description.

Example

```
Command: /applications/2

Response:
{"name":"Application 1","id":"2","description":""}
```

/applications/{appId}/environments

Retrieves all the environments that are linked to a specific application.

Resource URL

Type	URL and Format
GET	http://<host>:< port>/datamanagement/a/api/<version>/applications/{appId}/environments

Method Parameters

Parameters	Type	Description
appId	Long	Application ID

Response Parameters

Type	Description
EnvironmentApiDto (see page 158)	The environment dto. The fields are the environment name, environment id, application name, application id and description

Example:

```
Command: /applications/2/environments

Response:
{"name":"env2","id":"4","description":"Automatically
created","applicationId":"4","applicationName":"Myappname"}
```

/applications/{appId}/environments/{envId}

Retrieves specific environment details using the application id and the environment id in the path.

Resource URL

Type	URL and Format
GET	http://<host>:<port>/datamanagement/a/api/<version>/applications/{appId}/environments/{envId}

Method Parameters

Parameters	Type	Description
appId	Long	Application ID
envId	Long	Environment ID

Response Parameters

Type	Description
EnvironmentApiDto (see page 158)	The environment dto. The fields contain the environment name, environment id, application name, application id and description

Example

```
Command: /applications/2/environments/2

Response:
{"name":"env2","id":"4","description":"Automatically
created","applicationId":"4","applicationName":"Myappname"}
```

/applications/{appId}/environments/{envId}/releases

Retrieves all releases for a specific application and environment by entering the application id in the path.

Resource URL

Type	Description
GET	http://host:< port>/datamanagement/a/api/<version>/applications/{appId}/environments/{envId}/releases

Method Parameters

Parameters	Type	Description
appId	Long	Application ID
envId	Long	Environment ID

Response Parameters

Type	Description
ReleaseApiDto[] (see page 161)	The Dto contains the available data on a release

Example

```
Command: /applications/2/environments/2/releases

Response:
[{"name":"OnlineStore","id":"27","status":"Open","version":"3.0.16","description"
:"Release created by superuser through ROC
API","environmentId":"4","applicationId":"4","applicationName":"Myappname","envir
onmentName":"env2"},
{"name":"OnlineStore","id":"26","status":"Open","version":"3.0.15","description":
"Release created by superuser through ROC
API","environmentId":"4","applicationId":"4","applicationName":"Myappname","envir
onmentName":"env2"},
{"name":"OnlineStore","id":"29","status":"Open","version":"3.0.18","description":
"OnlineStore -
updated1","environmentId":"4","applicationId":"4","applicationName":"Myappname","
environmentName":"env2"}]
```

/applications/{appId}/environments/{envId}/releases/{releaseId}

Retrieves a specific application by entering the application id in the path.

Resource URL

Type	URL and Format
GET	http://<host>:< port>/datamanagement/a/api/<version>/applications/{appId} /environments/{envId}/releases/{releaseId}

Method Parameters

Parameters	Type	Description
appId	Long	Application ID

Parameters	Type	Description
envId	Long	Environment ID
releaseId	Long	Release ID

Response Parameters

Type	Description
ReleaseApiDto (see page 161)	The Dto contains the available release data

Example

```
Command: /applications/4/environments/4/releases/27

Response:
{"name":"OnlineStore","id":"27","status":"Open","version":"3.0.16","description":
"Release created by superuser through ROC

API","environmentId":"4","applicationId":"4","applicationName":"Myappname","envir
onmentName":"env2"}
```

/applications/{appId}/projects

Retrieves all active projects for a specific application

Resource URL

Type	Description
GET	http://<host>:< port>datamanagement/a/api/<version>/applications/{appId}

Method Parameters

Parameters	Type	Description
appId	Long	Application ID

Response Parameters

Type	Description
ProjectApiDto (see page 160)	The dto contains the name, description and application of a project.

Example

```
Command: /applications/2/projects

Response:
[{"archived":false,"applicationId":"2","name":"proj1","id":"1"},{"archived":false
,"applicationId":"2","name":"proj2","id":"21"}]
```

/applications/{appId}projects/{projectId}

Retrieves a specific project by providing the project id and the application id in the path.

Resource URL

Type	Description
GET	http://<host>:< port>datamanagement/a/api/<version>/applications/{appId}

Method Parameters

Parameters	Type	Description
appId	Long	Application ID
projectId	Long	project ID

Response Parameters

Type	Description
ProjectApiDto (see page 160)	The dto contains the name, description and application of a project.

Example

```
Command: /applications/{appId}/projects/{projectId}

Response: {"archived": "false","applicationId": "2","name":
"Version1.0","description": "TestVersion","id": "1"}
```

/applications/{appId}/projects/{projectId}/deployment-plans

Retrieves all deployment plans under application and project by entering the application and project ids in the path.

Resource URL

Type	Description
GET	http://<host>:< port>datamanagement/a/api/<version>/applications/{appId}

Method Parameters

Parameters	Type	Description
appId	Long	Application ID
projectId	Long	project ID

Response Parameters

Type	Description
DeploymentPlanResponseApiDto (see page 157)	The dto contains all the available values of a deployment plan.

Example

```
Command: /applications/{appId}/projects/{projectId}/deployment-plans
```

```
Response:
[{"result":true,"deploymentPlanId":"3","application":"app","project":"proj","deploymentPlan":"a","deploymentTemplate":"newDT","build":"a","templateCategory":"newTmp"},{"result":true,"deploymentPlanId":"1","application":"app","project":"proj","deploymentPlan":"newDP","artifactPackage":"artpack","deploymentTemplate":"newDT","build":"1.0","templateCategory":"newTmp"}]
```

/applications/{appId}/projects/{projectId}/deployment-plans/deploymentPlanId

Retrieves all deployment plans under application and project by entering the application and project ids in the path.In addition, status of all environments on which the deployments were created/executed.

Resource URL

Type	Description
GET	http://<host>:< port>datamanagement/a/api/<version>/applications/{appId}

Method Parameters

Parameters	Type	Description
appId	Long	Application ID
projectId	Long	project ID
deploymentPlanId	Long	Deployment plan ID

Response Parameters

Type	Description
DeploymentPlanResponseApiDto (see page 157)	The dto contains all the available values of a deployment plan.

Example

Command:
/applications/{appId}/projects/{projectId}/deployment-plans/{deploymentPlanId}

Response:
{"application":"app","project":"proj","result":true,"deploymentPlan":"newDP","artifactPackage":"artpack","deploymentTemplate":"newDT","build":"1.0","templateCategory":"newTmp","deploymentsStatus":[{"status":"PENDING","environment":"Environment for Default Architecture","deployment":"run_deployment","deploymentId":"1"},{"status":"PENDING","environment":"Environment for Default Architecture","deployment":"run_deployment2","deploymentId":"2"},{"status":"PENDING","environment":"Environment for Default Architecture","deployment":"run_deployment3","deploymentId":"3"},{"status":"SUCCEEDED","environment":"Environment for Default Architecture","deployment":"delivers","deploymentId":"10"}],"deploymentPlanId":"1"}

/applications/{appId}/templates

Retrieves all templates for a specific application using the application id in the path.

Resource URL

Type	Description
GET	http://<host>:< port>/datamanagement/a/api/<version>/applications/{appId}/templates

Method Parameters

Parameter	Type	Description
appId	Long	Application ID

Response Parameters

Type	Description
TemplateApiDto (see page 166)	The Dto contains the available template data

Example

```
Command: /applications/2/templates

Response:
[{"name":"Temp
2","id":"12","description":"","applicationId":"4","applicationName":"My
Application"},{"name":"Temp
1","id":"11","description":"","applicationId":"4","applicationName":"My
Application"},{"name":"temp3","id":"13","description":"","applicationId":"4","app
licationName":"My App"}]
```

/applications/{appId}/templates/{templateId}

Retrieves a specific template using the application id and the template id in the path.

Resource URL

Type	Description
GET	http://<host>:< port>/datamanagement/a/api/<version>/applications /{appId}/templates/{templateId}

Method Parameters

Parameters	Type	Description
appId	Long	Application ID
templateId	Long	Template ID

Response Parameters

Type	Description
TemplateApiDto (see page 166)	The Dto contains all the available template data

Example

```
Command: /applications/2/templates/12

Response:
{"name":"T2","id":"12","description":"","applicationId":"4","applicationName":"My
appname"}
```

/artifact-details

Retrieves the details of an artifact in one of the following way:

- Artifact ID {artifactId}
- Application ID {applicationId}, Artifact name {artifact} and artifact version {artifactVersion}
- Application name {application}, Artifact name {artifact} and artifact version {artifactVersion}

Note: The API is relevant only for v4.7. From v5.0, use /artifact-version-details

Resource URL

Type	URL and Format
POST	http://<host>:< port>/datamanagement/a/api/<version>/artifact-details

Request Parameters

Type	Description
ArtifactBasicApiDto (see page 148)	The basic artifact dto. To specify an artifact uses either the artifact id or the artifact name, version, and either the application name or id. All fields are in the dto

Response Parameters

Type	Description
ArtifactsApiDto (see page 147)	This Dto contains all the available data on an artifact

Example

```
Command: /artifact-details

Body:
{"artifactId":"4"} or {"applicationId":"8","artifact":"New artifact from
REST","version":"1.0.1"}

Response:
{"properties":{"Url":"http://Harta","Password":null,"Username":null,"File
alias":"unnamedFile1374096003201"},"version":"1.0.1","applicationId":"8","allowAr
tifactModifications":true,
"artifactId":"1","getterType":"HTTP","storeInRepository":false,"artifactName":"Bl
a1 bcla new artifact from REST"}
```

/artifact-status

Retrieves the status of an artifact in one of the following ways:

- Artifact ID {artifactId}
- Application ID {applicationId}, Artifact name {artifact} and artifact version {artifactVersion}
- Application name {application}, Artifact name {artifact} and artifact version {artifactVersion}

Note: The API is relevant only for v4.7. From v5.0, use /artifact-version-status.

Resource URL

Type	URL and Format
POST	http://<host>:< port>/datamanagement/a/api/<version>/artifact-status

Request Parameters

Type	Description
ArtifactBasicApiDto (see page 148)	The basic artifact dto. To specify an artifact uses either the artifact id or the artifact name and version and either the application name or id. All fields a the dto

Type	Description
ArtifactStatusApiDto (see page 150)	Retrieves the data about the status of the artifact. If the request fails, the d contains information regarding the failure.

Example

```
Command: /artifact-status

Body:
{"artifactId":"4"} or {"applicationId":"8","artifact":"New artifact from
REST","version":"1.0.1"}

Response:
{"id":"4","status":"NOT_IN_REPOSITORY","description":"NOT_IN_REPOSITORY","result"
:true}
```

/artifact-version-details

Retrieves the details of an artifact-version in one of the following ways:

- Artifact-version id
- Artifact-definition id and artifact-version name
- Application name, artifact-type name, artifact-definition name and artifact-version

(V3+)

Resource URL

Type	URL and Format
POST	http://<host>:< port>/datamanagement/a/api/<version>/artifact-version-details

Request Parameters

Type	Description
ArtifactVersionsApiDto (see page 150)	Contains the location of a specific artifact version

Response Parameters

Type	Description
ArtifactsApiDto (see page 147)	This Dto contains all the available data on an artifact

Example

```
Command: /artifact-version-details
Body:
{"artifactId":"4"} or {"artifactDefinitionId":"8","version":"1.0.1"} or
{"application":"App1","artifactType":"jar","artifactDefinition":"travel","version
":"1.0.1"}
Response:
{"properties":{"Url":"http://Harta","Password":null,"Username":null,"File
alias":"unnamedFile1374096003201"},"version":"1.0.1","applicationId":"8","allowAr
tifactModifications":true,
"artifactId":"1","getterType":"HTTP","storeInRepository":false,"artifactName":"Bl
a1 bcla new artifact from REST"}
```

/artifact-version-status

Retrieves the upload status of an artifact in one of the following ways:

- Artifact-version id
- Artifact-definition id and artifact-version name
- Application name, artifact-type name, artifact-definition name and artifact-version

(V3+)

Resource URL

Type	URL and Format
POST	http://<host>:< port>/datamanagement/a/api/<version>/artifact-version-status

Request Parameter

Type	Description
ArtifactVersionApiDto (see page 150)	Contains the location of a specific artifact version

Response Parameter

Type	Description
ArtifactStatusApiDto (see page 150)	Retrieves data regarding the status of the artifact. if the request fails, the data will contain information regarding the failure.

Example

```
Command: /artifact-version-status
Body:
{"artifactId":"4"} or {"artifactDefinitionId":"8","version":"1.0.1"} or
{"application":"App1","artifactType":"jar","artifactDefinition":"travel","version
":"1.0.1"}
Response:
{"id":"4","status":"NOT_IN_REPOSITORY","description":"NOT_IN_REPOSITORY","result"
:true}
```

/assign-new-artifact-package-to-deployment-plan

Creates and Adds an artifact package to an existing deployment plan.

The required parameters are the deployment plan, project, build, and application names, with the xml for the artifact package.

Resource URL

Type	Description
POST	http://<host>:< port>datamanagement/a/api/<version>/assign-new-artifact-package-to-deployment-plan

Request Parameters

Type	Description
CreateArtifactFroDeploymentPlanDto (see page 152)	The dto contains the fields to create and assign the artifact package to deployment plan

Response Parameters

Type	Description
ResponseDataApiDto (see page 163)	The general Dto that is returned to the user. The data in the dto is concurrent with the success or the failure of the request.

Example

```
Command: /assign-new-artifact-package-to-deployment-plan

Body:

{"deploymentPlanId":"154", "applicationId":"2","xml":"<package
xmlns="http://www.example.org/ArtifactPackage"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.example.org/ArtifactPackage package.xsd">
<description>A very special </description> <name>My new Package</name>
<artifactVersionByPath> <definition>travel</definition> <type>war</type>
<version>1</version> </artifactVersionByPath> </package>"}

Response:

{"description":"Successfully created an artifact package with id [22] and assigned
to deployment plan with id [154].","result":true}
```

/create-artifact

Creates an artifact with the different parameters.

Note: The API is relevant for only v4.7. From v5.0, use /create-artifact-package-xml.

Resource URL

Type	Url and Format
POST	http://<host>:< port>/datamanagement/a/api/<version>/create-artifact

Request Parameters

Type	Description
CreateArtifactApiDto (see page 151)	The Dto contains all the available data needed to create an artifact

Response Parameters

Type	Description
ResponseData (see page 163)	The Dto that is returned to the user. The data in the dto is concurrent with the success or the failure of the request.

Example

```
Command: /create-artifact

Body:

{"application":"Myappname","server":"127.0.0.1","description":"my
desc","artifact":"new artifact from
REST","version":"v1.2.1","allowArtifactModifications":"true","getterType":"HTTP",
"properties":{"Url":"http://someurl","File alias":"1111"}}

Response:

{"id":"23","description":"Artifact created successfully","result":true}
```

/create-artifact-package

Creates an empty artifact package (V3+)

Resource URL

Type	URL and Format
POST	http://<host>:< port>/datamanagement/a/api/<version>/create-artifact-package

Request Parameter

Type	Description
CreateArtifactPackageApiDto (see page 152)	Contains the name, description, and application of the artifact package

Response Parameter

Type	Description
ResponseData (see page 163)	This is the general Dto that is returned to the user. The data in this dto is concurrent with the success or the failure of the request.

Example

```
Command: /create-artifact-package
Body:
{"name":"Singapore Travel", "applicationId":"2"}
Response:
{"result":"true", "description":"Artifact-package created successfully"}
```

/create-artifact-package-xml

Creates a new artifact package provided in XML (V3+)

Resource URL

Type	URL and Format
POST	http://<host>:< port>/datamanagement/a/api/<version>/create-artifact-package-xml

Request Parameter

Type	Description
ArtifactPackageUploadDto (see page 149)	Contains the xml of Artifact Package

Response Parameter

Type	Description
ResponseData (see page 163)	The general Dto that is returned to the user. The data in the dto is concurrent with the success or the failure of the request.

Example

```
Command: /create-artifact-package-xml
Body:
{"xml":"< ... >"}
Response:
{"result":"true", "id":"12", "description":"Artifact-package created successfully"}
```

/create-artifact-version

Creates an artifact-version. Supports creation by artifact-definition id or by application id, artifact-type name and artifact-definition name.

Resource URL

Type	URL and Format
POST	http://<host>:< port>/datamanagement/a/api/<version>/create-artifact-version

Request Parameter

Type	Description
CreateArtifactVersionApiDto (see page 153)	This Dto contains all the avaliable data needed to create an artifact-version

Response Parameter

Type	Description
ResponseData (see page 163)	This is the general Dto that is returned to the user. The data in this dto is concurrent with the success or the failure of the request.

Example

```
Command: /create-artifact-version
Body:
{"artifactDefinitionId":"2","server":"127.0.0.1","description":"my
desc","artifact":"new artifact from
REST","version":"v1.2.1","allowArtifactModifications":"true","getterType":"HTTP",
"properties":{"Url":"http://someurl","File alias":"1111"}}
Response:
{"id":"23","description":"Artifact created successfully","result":true}
```

/create-deployment-plan

Creates a deployment plan from an existing deployment template. An artifact package is assigned to the deployment plan. An existing artifact package (by name), or a new one (if supplied an XML). In addition, the deployment plan can load a manifest file if supplied one.

Note: A project is created when not found.

Resource URL

Type	Description
POST	http://<host>:< port>datamanagement/a/api/<version>/create-deployment-plan

Request Parameters

Type	Description
DeploymentPlanApiDto (see page 156)	The dto contains the fields used to create a deployment plan.

Response Parameters

Type	Description
DeploymentPlanResponseApiDto (see page 157)	The dto contains the available values of a deployment plan.

Example

```
Command: /create-deployment-plan

Body:

Example 1:
{"deploymentPlan":"deployREST","build":"buildREST","project":"newProj","deploymen
tTemplate":"newDeployment","templateCategoryId":"36","application":"app"}

Example 2:
{"deploymentPlan":"deployREST2","build":"buildREST2","project":"newVersion","depl
oymentTemplate":"newDeployment","templateCategory":"newTemplate","application":"a
pp"}

Example 3:
{"deploymentPlan":"newDP5","build":"2.0","project":"proj","deploymentTemplate":"n
ewDT","templateCategory":"newTmp","application":"Parameters test",
"artifactPackage":"art pack"}

Response:
{"result":true,"deploymentPlan":"newDP5","deploymentPlanDescription":"Successfull
y created a deployment plan [newDP5] with id [27]","deploymentPlanId":"27"}
```

/create-project

Creates a new project for a given application

Resource URL

Type	URL and Format
POST	http://<host>:< port>/datamanagement/a/api/<version>/creat e-project

Request Parameter

Type	Description
ProjectApiDto (see page 160)	The dto contains the name, description and application of a project.

Response Parameter

Type	Description
ResponseData (see page 163)	The general Dto that is returned to the user. The data in the dto is concurrent with the success or the failure of the request.

Example

```
Command: /create-project
Body:
Example 1: {"name":"Version 3.0","description":"TestVersion3","applicationId":"1"}
Response:
{id: "12","description": "Project Version 3.0 created successfully","result":
"true"}
```

/create-release

Creates a release from an existing template.

Set the parameters in the JSON object in one of the following ways:

- Use application {application name}, template {template name}
- Use template id {templateid}

The release details are:

- Use release {release name}, environment {environment name}, release type {release type – Minor/ Major/Emergency. Default is Minor}, doStepsValidation {true/false. Whether to perform step validation. If one step does not match the specified environment, fail the operation. Default is true.}

Note: The create release operation runs synchronously.

Note: The API is relevant only for v4.7. From v5.0, use /run-deployment-plan

Resource URL

Type	URL and Format
POST	http:/<host>:< port>/datamanagement/a/api/<version>/create-release

Request Parameters

Type	Description
CreateReleaseApiDto (see page 154)	The Dto contains the data that specifies to create a release from a template - individualize a template and define basic release entities: name, version, description, type, environment and if to validate the steps of the environment before the run.

Response Parameters

Type	Description
ResponseData (see page 163)	The general Dto that is returned to the user. The data in the dto is concurrent with the success or the failure of the request.

Example

```
Command: /create-release

Body:

Example 1: {"environment":"staging","template":"OnlineStore
Template","release":"REST
API","application":"OnlineStore","version":"1.0.1","doStepsValidation":"false","r
eleaseType":"Major"}

Example 2: {"environment":"env2","templateId":"26","release":"REST
API","version":"1 - beta"}

Response:

{"id":"47","description":"Release REST API created successfully","result":true}
```

/delete-release

Delete a release using basic release parameters.

Set the parameters in the JSON object in one of the following ways:

- Use application {application name}, environment {environment name}, release {release name} and version {version name}
- Use the release id {releaseId}.

Note: The stop operation runs synchronously.

Resource URL

Type	URL and Format
POST	http://<host>:< port>/datamanagement/a/api/<version>/delete-release

Request Parameters

Type	Description
ReleaseBasicApiDto (see page 162)	The Dto that contains the basic fields that specify a release. For more information, see the *CA Release Automation API Reference Guide 4.7*.

Response Parameters

Type	Description
ResponseData (see page 163)	The general Dto that returns to the user. The data in the Dto is concurrent with the success or failure of the request.

Example:

```
Command: /delete-release

Body:
{"application":"Myappname","release":"REST API 4","environment":
"production","version":"1.1"}
{"releaseId":"83"}

Response:
{"id":"27","description":"Release with ID [83] was deleted.","result":true}
```

Note: Access the full REST API documentation at http://<hostname>:8080/datamanagement/restApi.jsp where hostname is the location of your Management Server.

/export-release

Use a release ID to export a release

Resource URL

Type	URL and Format
POST	http://<host>:< port>/datamanagement/a/api/<version>/export-release

Request Parameters

Type	Description
ReleaseBasicApiDto (see page 162)	The dto contains the basic fields for specifying a release.

Example

```
Command: /export-release

Body:

{"releaseId":"83"}

Response:

<exportData>
     <details>
      <user>superuser</user>
      <version>04.7.004.7.0.148</version>
      <date>20/06/2013</date>
      <time>15:15:54</time>
      <exportType>release</exportType>
    </details>
    <release name="rel1" version="1.0.1" environment="ENV1" application="Rest">
      <description>Release created by superuser through ROC API</description>
      <type>Major</type>
      <status>Initialization Failed</status>
      <template>Init failed</template>
      <startTime>10:09:22 16/06/2013</startTime>
      <endTime>10:09:22 16/06/2013</endTime>
      <properties>
        <property name="Release Version">1.0.1</property>
        <property name="Application Name">Rest</property>
        <property name="Release Name">rel1</property>
        <property name="Environment Name">ENV1</property>
      </properties>
      <initialization>
         <environment>ENV1</environment>
         <startTime>08:54:51 13/06/2013</startTime>
         <endTime>10:09:21 16/06/2013</endTime>
         <step name="Initialization Step"
    rollbackImpact="triggerRollbackOnFailure">
          <process tag="latest">Fail Compare</process>
          <status>Fail</status>
          <server-type name="Server Type 1">
           <servers useAllServers="false">
             <server name="MJ" ip="192.168.0.26" nodeId="MJ0"/>
           </servers>
         </server-type>
       </step>
     </initialization>
     <steps>
      <step name="delay " rollbackImpact="triggerRollbackOnFailure">
       <process tag="latest">delay</process>
       <status>Canceled</status>
       <server-type name="Server Type 1">
```

```
    <servers useAllServers="false">
    <server name="MJ" ip="192.168.0.26" nodeId="MJ0"/>
   </servers>
  </server-type>
   <server-type name="Server Type 2">
    <servers useAllServers="false"/>
  </server-type>
  </step>
  </steps>
   <post-deployment/>
  </release>
 </exportData>
```

/export-template

Exports a template. The result is a JSON object that contains an XML format of the template data

Resource URL

Type	URL and Format
POST	http://<host>:< port>/datamanagement/a/api/<version>/export-template

Request Parameters

Type	Description
TemplateBasicApiDto (see page 167)	The dto contains the basic fields for specifying a template.

Example

```
Command: /export-template

Response:
    <exportData>
      <details>
        <user>superuser</user>
        <version>04.8.0.1</version>
        <date>23/09/2013</date>
        <time>14:04:38</time>
        <exportType>template</exportType>
      </details>
      <template name="t" application="Json">
        <description></description>
```

```
    <status>locked</status>
    <properties>
      <property name="Release version"></property>
      <property name="Application Name">Json</property>
      <property name="Release Name">t</property>
      <property name="Environment Name"></property>
      <property name="Release Name">t</property>
      <property name="Environment Name"></property>
    </properties>
    <initialization/>
    <steps/>
    <post-deployment>
  </template>
</exportData>
```

/export/processes

Export process details to a PDF document using specific filters

Resource URL

Type	URL and Format
GET	http://<host>:< port>/datamanagement/a/api/<version>/export/processes

Query Parameters

Parameters	Type	Description
reportName	String	Report name
reportDescription	String	Report description
type	String	Type filter
period	Integer	Period
applications	List	Application filter
environments	List	Environment filter
categories	List	Categories filter
processes	List	Processes filter
users	List	Users filter
statusType	List	Status type filter
templates	List	Template filter
versions	List	Versions filter

Parameters	Type	Description
statuses	List	Statuses filter

Response Parameters

Type	Descritption
ResponseData (see page 163)	The general Dto that is returned to the user. The data in the dto is concurrent with the success or the failure of the request.

Example

```
Command:
/export/processes?type=raw+data&reportName=example&reportDescription=example+of+e
xport-processes&period=100
```

/export/releases

Export a release details to a PDF document using specific filters.

Resource URL

Type	URL and Format
GET	http://<host>:< port>/datamanagement/a/api/<version>/export/releases

Query Parameters

Parameters	Type	Description
reportName	String	
reportDescription	String	
type	String	
period	Integer	
applications	List	
environments	List	
categories	List	
processes	List	
users	List	
statusType	List	
templates	List	

Parameters	Type	Description
versions	List	
statuses	List	

Response Parameters

Type	Description
ResponseData (see page 163)	The general Dto that is returned to the user. The data in the dto is concurrent with the success or the failure of the request.

Example

```
Command:
export/releases?type=raw+data&reportName=example&reportDescription=example+of+exp
ort-releases&period=100
```

/get-artifact-package-content

Retrieves artifact-versions for a given artifact-package (V3+)

Resource URL

Type	URL and Format
Post	http://<host>:< port>/datamanagement/a/api/<version>/get-artifact-package-content

Request Parameters

Type	Description
ArtifactPackageApiDto (see page 149)	Contains the ID or path of an artifact-package

Response Parameters

Type	Description
ArtifactNameDTOList (see page 149)	

Example

```
Command: /get-artifact-package-content

Body:
{"artifactPackageId":"2"}
Response:
{"list":[{"name":"Travel.2.0.12.war","description":"Beta version of Travel
app","id":"12"}]}
```

/get-artifact-versions

Retrieves artifact-versions for a given artifact-definition (V3+)

Resource URL

Type	URL and Format
POST	http://<host>:< port>/datamanagement/a/api/<version>/get-artifact-versions

Request Parameters

Type	Description
ArtifactVersionsApiDto (see page 150)	

Response Parameters

Type	Description
ArtifactNameDTOList (see page 149)	

Example

```
Command: /get-artifact-versions
Body:
{"artifactDefinitionId":"2"}
Response:
{ "list" : [ {"name" : "Travel.2.0.12.war"} ] }
```

/get-environment-parameter

Retrieves the value of a parameter per environment.

Result is a JSON object containing the values of the parameter (simple, array or loop folder value).

Resource URL

Type	URL and Format
POST	http://<host>:< port>/datamanagement/a/api/<version> /get-environment-parameter

Request Parameters

Type	Description
EnvironmentParameterApiDto (see page 159)	This dto contains the basic fields that describe a parameter for a specific environment.

Request Parameters

Type	Description
ReponseEnvironmentParameterApiDto (see page 163)	This dto contains the fields that describe the value of parameter.

Example

```
Command: /get-environment-parameter
Body:
Example 1: {"environmentId":"3","parameterPath":"Default
Component/Production/foldersArray"}
Example 2: {"applicationId":"1","environment":"Dev","parameterPath":"Default
Component/Production/foldersArray"}
Response:
{"parameterPath":"Default
Component/Production/foldersArray","arrayValue":["GDAY","MATE"]}
```

/load-manifest

Loads a manifest file to a deployment plan.The required fields are the deployment plan Id and the manifest xml.

Resource URL

Type	URL and Format
POST	http://<host>:< post>/datamanagement/a/api/<version> /load-manifest

Request Parameters

Type	Description
LoadManifestApiDto (see page 160)	The dto contains the fields used to load a manifest file to a deployment plan.

Request Parameters

Type	Description
ResponseDataApiDto (see page 163)	The general Dto that is returned to the user. The data in the dto is concurrent with the success or the failure of the request.

Example

```
Command: /load-manifest
Body:
Example: {"deploymentPlanId": "320","manifest": "<DeploymentTemplate name="
deployment template num_1392197922873 " template="template_num_1392197922799 "
application="Parameters test">
<Initialization/>
<Deployment>
<step name=" Step-1392197923088">
<Parameters_Scope_Process>
<Application_Parameters>
<str_env_parameter>my manifest value 1392197923824</str_env_parameter>
</Application_Parameters>
</Parameters_Scope_Process>
</step>
</Deployment>
<Post-Deployment/>
</DeploymentTemplate>"}
Response:
{description":"Successfully loaded manifest file.","result":true}
```

/release-status

Uses the basic release details in the body to retrieve the status of a release. The required parameters in the JSON object are set in one of following ways:

- Use application {application name}, environment {environment name}, release {release name} and version {version name}
- Use {releaseId} release id.

Resource URL

Type	URL and Format
POST	http://<host>:< port>/datamanagement/a/api/<version>/release-status

Request Parameters

Type	Description
ReleaseBasicApiDto (see page 162)	The basic release dto. To specify a release use either the release id or the release name, release version, application name and application id. All fields are in the dto

Response Parameters

Type	Description
ReleaseStatusApiDto (see page 162)	The Dto contains two sections: Success and Failure, not all fields appear in both sections Success - all the fields contain data linked to the release Failure - all the fields contain data about the reason of the failure

Example

```
Command: /release-status

Body:
{"application":"Myappname","release":"REST API 4","environment":
"production","version":"1.1"}
 {"releaseId":"83"}

Response:
v1: {"id":"16","description":"100%
Succeeded","result":true,"stage":"Post-Deployment","releaseStatus":"Succeeded","s
tageState":"Succeeded"}
v2: {id: "16" description: "100% Succeeded"result: truestage:
"Post-Deployment"releaseStatus: "Succeeded"stageState: "Succeeded"}
```

/release-status/{releaseId}

Retrieves the status of a release using the release id in the path. Same result as "/release-status" with POST protocol mentioned above. Used instead of "release-status" when the release name contains digits.

Response URL

Type	URL and Format
GET	http://<host>:< port>/datamanagement/a/api/<version>/release-status/{releaseId}

Method Parameters

Parameters	Type	Description
releaseId	Long	The basic release dto. To specify a release use either the release id or the release name, release version, application name and application id. All fields are in this dto.

Response Parameters

Type	Description
ReleaseStatusApiDto (see page 162)	The Dto divides into two sections: Success and Failure, and not all fields appear in both sections. Success - all the fields contain data linked to the release Failure - all the fields contains the reason of failure data

Example

```
Command: /release-status/23

Response:
v1: {"id":"16","description":"100%
Succeeded","result":true,"stage":"Post-Deployment","releaseStatus":"Succeeded","s
tageState":"Succeeded"}
v2: {id: "16" description: "100% Succeeded"result: truestage:
"Post-Deployment"releaseStatus: "Succeeded"stageState: "Succeeded"}
```

/releases-reports

Retrieves all releases by using different filters in the url. Users may add a few attributes to a filter, such as /releases-reports?application=2&application=3. This retrieves the releases from both applications. Also in the 3 filters: application, environment and template, specify either the name or the id of the filter. i.e. application=application1 and application=2.

Resource URL

Type	URL and Format
GET	http://<host>:< port>/datamanagement/a/api/<version>/releases-reports

Query Parameters

Parameters	Type	Description
template	String	Template ID or template name
application	String	Application ID or application name
environment	String	Environment ID or environment name
releaseStatus	List	List of release statuses, each value can be entered as a single value. i.e. &releaseStatus=running&releaseStatus=succeeded
stage	List	List of stages that the release can be in, each value can be entered as a single value. i.e. &stage=deployment&stage=post-deployment
stageState	List	List of states that the release stages can be in. It is relevant only for v2. The list of available stages are: Pending/Running/Paused/Running-With-Errors/Succeeded/Failed/Canceled. Each value can be entered as a single value. i.e. &stageState=&stageState=succeeded
period	Integer	

Response Parameters

Type	Description
ReleaseApiDto[] (see page 161)	The Dto contains the available data on a release.

Examples:

The Command:
http://localhost:8080/datamanagement/a/api/v2/releases-reports?releaseStatus=Active&application=2&environment=env1&stageState=pending&stage=deployment&period=12

The Response:
{"version":"","templateName":"dealy","applicationName":"Rest","endTime":"","status":"0% Pending","stage":"Deployment","startTime":"","releaseStatus":"Active","applicationId":"2","environmentId":"2","templateId":"3","environmentName":"ENV1","stageState":"Pending","name":"rel","description":"","id":"9"},{"version":"","templateName":"t","applicationName":"Rest","endTime":"","status":"0% Pending (With errors)","stage":"Deployment","startTime":"","releaseStatus":"Active","applicationId":"2","environmentId":"2","templateId":"4","environmentName":"ENV1","stageState":"Pending","name":"rel14","description":"","id":"10"}

/run-deployment-plan

Creates a deployment plan from an existing deployment template.A artifact package could be assigned to the deployment plan. An existing artifact package (by name), or a new one (supplied by XML). The deployment plan is also assigned an artifact package, when supplied the artifact package name. The deployment plan can also load a manifest file when supplied one.

Note: A project is created if not found.

Resource URL

Type	URL and Format
POST	http:/<host>:< port>/datamanagement/a/api/<version>/run-deployment-plan

Request Parameters

Type	Description
DeploymentPlanApiDto (see page 156)	The dto contains the fields used to create a deployment plan

Response Parameters

Type	Description
DeploymentPlanResponseApiDto (see page 157)	The dto contains the available values of a deployment plan

Example

```
Command: /run-deployment-plan

Body:
{"deploymentPlan":"deployREST151","build":"buildREST2","project":"newVersion","deploymentTemplate":"deploymentTemplate","templateCategory":"newTmp","application":"Parameters test","deployment":"run_deployment61", "environments":["Environment for Default Architecture"],"deploymentStageToPerform":"none"}

Response:
{"result":true,"deploymentResults":[{"envName":"Environment for Default Architecture","envId":"4","id":"9","description":"Successfully created deployment [run_deployment61] with id [9] on environment [Environment for Default Architecture] with id [4]","result":true}],"deploymentPlanDescription":"Successfully created a deployment plan [deployREST151] with id [74]","deploymentPlan":"deployREST151","deploymentPlanId":"74"}
```

/run-deployments

Creates a deployment from an existing deployment plan. Users can create or run the deployment on the environments provided.

Retrieve the status of a step using the step id in the body.

Resource URL

Type	URL and Format
POST	http://<host>:< post>/datamanagement/a/api/<version>/run-deployments

Request Parameters

Type	Description
DeploymentApiDto (see page 155)	The dto contains the fields to create and/or run a release deployment.

Response Parameters

Type	Description
DeploymentResponseApiDto (see page 156)	The dto contains the available values of a deployment.

Example

```
Command: run-deployments

Body:
Example 1: {"deployment":"run_deployment31","application":"app",
"environments":["Environment for Default Architecture", "env2", "env3"],
"deploymentPlan":"newDP", "build":"1.0", "project":"proj",
"stageToPerform":"none"}
Example 2: {"deployment":"run_deployment32","application":"app",
"environments":["Environment for Default Architecture", "env2", "env3"],
"deploymentPlan":"newDP", "build":"1.0", "project":"proj",
"stageToPerform":"Deployment"}


Response:
Example 1: [{"envName":"env3","envId":"5","id":"81","description":"Successfully
created deployment [run_deployment31] with id [81] on environment [env3] with id
[5]","result":true},
{"envName":"env2","envId":"4","id":"80","description":"Successfully created
deployment [run_deployment31] with id [80] on environment [env2] with id
[4]","result":true},
{"envName":"Environment for Default
Architecture","envId":"2","id":"79","description":"Successfully created deployment
[run_deployment31] with id [79] on environment [Environment for Default Architecture]
with id [2]","result":true}]

Example 2: [{"id":"84","description":"Deployment [run_deployment32] with id [84] has
started runnning on environment [env3].","result":true},
{"id":"83","description":"Deployment [run_deployment32] with id [83] has started
runnning on environment [env2].","result":true},
{"id":"82","description":"Deployment [run_deployment32] with id [82] has started
runnning on environment [Environment for Default Architecture].","result":true}]
```

/run-release

Run a prepared release by individualizing a release () by entering

The Result contains the following fields:

- ID. ID of newly created release. If creation failed, ID is null.

- Description. Error or success message describing the result of the operation.
- Result. Return of true indicates creation succeeded; return of false indicates creation failed

Note: The API is relevant only for v4.7. From v5.0, use /run-deployments

Resource URL

Type	URL and Format
POST	http://<host>:< port>/datamanagement/a/api/<version>/run-release

Request Parameters

Type	Description
RunReleaseApiDto (see page 164)	The dto contains the data required to run a release - individualize the release and define the timeout, if to run asynchronously or not.

Response Parameters

Type	Description
ResponseData (see page 163)	The general Dto that is returned to the user. The data in the dto is concurrent with the success or the failure of the request.

Examples:

```
The Command:
http://localhost:8080/datamanagement/a/api/run-release
{"application":"OnlineStore","environment": "production","release":"REST
API","version":"1.0.1","asynch":"true","timeout":"60"}
{"releaseId":"7","asynch":"true"}

The Response:

{"id":"48","description":"Starting release...","result":true}
```

/run-template

Combination of {create-release, update-release, run-release}, with some limitations:

- The release runs asynchronously.
- The update release updates only release properties.

- The release properties update the properties before the init-step run only if the init-step page "Allow Modifications Before Release Run" flag is turned on. If not, the release properties are updated after the init-step finishes.
- Even if the result is true, it does not mean that the release started since the execution is made asynchronous.

Resource URL

Type	Description
POST	http://<host>:< port>/datamanagement/a/api/<version>/run-template

Request Parameters

Type	Description
RunTemplateApiDto (see page 165)	The dto contains the data required to run a template that includes: create release, update release and run release.

Response Parameters

Type	Description
ResponseData (see page 163)	The general Dto that is returned to the user. The data in the dto is concurrent with the success or the failure of the request.

Examples:

```
The Command: /run-template

Body:

{"templateId":"10","release":"Release name","environment":"env2","version":"1.6.8
","description":"My description","do
StepsValidation":"false","properties":{"a":"b"}}

The Response:
{"id":"197","description":"Release a started","result":true}
```

/schedule-release

Schedules a defined using the date, time, and duration of the scheduled release.

Scheduled dates and times use the CA Release Automation server's time zone and clock.

Resource URL

Type	URL and Format
POST	http://<host>:< port>/datamanagement/a/api/<version>/schedule-release

Request Parameters

Type	Description
ScheduleReleaseApiDto (see page 166)	The dto contains the data required to schedule a release - individualize a release and define the scheduled date, time and duration.

Response Parameters

Type	Description
ResponseData (see page 163)	The general Dto that is returned to the user. The data in the dto is concurrent with the success or the failure of the request.

Examples:

```
The Command: /schedule-release
Body:
{"application":"Online Store","environment": "env2","release":"REST
API","version":"1.0.1","scheduledDate":"24/01/13",
"scheduledTime":"12:00","estimatedDurationMinutes":"90"}
{"releaseId":"37","scheduledDate":"24/1/13",
"scheduledTime":"13:00","estimatedDurationMinutes":"90"}

The Response:
```

{"id":"37","description":"Release [37] scheduled for [1359025200000]. Estimated duration is [5400000].","result":true}

/step-status

Retrieve the status of a step using the step id in the body.

Resource URL

Type	URL and Format
POST	http://<host>:< port>/datamanagement/a/api/<version>/step-status

Request Parameters

Type	Description
StepApiDto (see page 166)	The dto contains just the step ID.

Response Parameters

Type	Description
ResponseData (see page 163)	The general Dto that is returned to the user. The data in the dto is concurrent with the success or the failure of the request.

Example

```
Command: /step-status

Body:
{"stepId":"23"}

Response:
{"id":"23","result":true,"description":"0% New"}
```

/step-status/{stepId}

Retrieves a step by entering the step id in the path.

Response URL

Type	Description
GET	http://<host>:< port>/datamanagement/a/api/<version>/step-status/{stepId}

Method Parameters

Parameters	Type	Description
stepId	Long	step ID

Response Parameters

Type	Description
ResponseData (see page 163)	The general Dto that is returned to the user. The data in this dto is concurrent with the success or the failure of the request.

Example

```
Command: /step-status/23

Response:
{"id":"23","result":true,"description":"0% New"}
```

/stop-release

Stops a run release using the release details.The required parameters in the JSON object are set in one of following ways:

- Use application {application name}, environment {environment name}, release {release name} and version {version name},
- Use release id {releaseId}.

Resource URL

Type	URL and Format
POST	http://<host>:< port>/datamanagement/a/api/<version>/stop-release

Request Parameters

Type	Description
ReleaseBasicApiDto (see page 162)	The basic release dto. To specify a release use either the release id or the release name, release version, application name, and application id. All fields are in the dto

Response Parameters

Type	Description
ResponseData (see page 163)	The general Dto that is returned to the user. The data in the dto is concurrent with the success or the failure of the request.

Example:

```
The Command: /stop-release
Body:
{"application":"Myappname","release":"REST API 4","environment":
"production","version":"1.1"}
{"releaseId":"83"}

The Response:

{"id":"50","description":"Stopping release...","result":true}
```

/update-environment-parameter

Retrieves the value of a parameter per environment.

Resource URL

Type	URL and Format
POST	http://<host>:< port>/datamanagement/a/api/<version>/update-environment-parameters

Request Parameter

Type	Description
FullEnvironmentParameterApiDto (see page 159)	This dto contains all the fields that are needed to update the value of an environment parameter

Response Parameter

Type	Description
ResponseDataApiDto (see page 163)	This is the general Dto that is returned to the user. The data in this dto is concurrent with the success or the failure of the request.

Example

```
Command: /update-environment-parameter
Body:
Example 1 (Simple Value): {"environmentId":"2","parameterPath":"Default
Component/p1","simpleValue":"Michael"}
Example 2 (Array Value): {"environment":"Production","parameterPath":"Default
Component/arr1","application":"App","arrayValue":["Gday","Mate","how","are","you"
,"doing?"]}
Example 3 (Loop Folder): {"environment":"Production","parameterPath":"Default
Component/AMQ","application":"App","loopValue": [{"instance.name": {"simpleValue":
"name1"},"location": {"simpleValue": "c:/temp"}},{"instance.name": {"simpleValue":
"name2"},"location": {"simpleValue": "c:/local/2"},"instance.name":
{"simpleValue": "name3"},"location": {"simpleValue": "C;/temp3"}}]}
Response:
Example 1 (Simple Value): {"result":true,"description":"The value for the parameter
[Default Component/p1] is [Michael]"}
Example 2 (Array Value): {"result":true,"description":"The value for the parameter
[Default Component/arr1] is [[Gday, Mate, how, are, you, doing?]]"}
Example 3 (Loop Folder): todo
```

/update-release

Update a release using XML to describe the changes. The required parameters in the JSON object are the release details and the XML. Release details are set in one of the following ways:

- Use application name, environment name, release name and version name.
- Use the release id.

Note: The API is relevant only for v4.7. From v5.0, use /load-manifest

Resource URL

Type	Description
POST	http://<host>:< port>/datamanagement/a/api/<version>/update-release

Request Parameters

Type	Parameters
UpdateReleaseApiDto (see page 168)	The dto contains the data required to update a release - individualize a release and the xml that describes the updates.

Response Parameters

Type	Description
ResponseData (see page 163)	The general Dto that is returned to the user. The data in the dto is concurrent with the success or the failure of the request.

Examples:

```
Command: /update-release
Body:
{"releaseId":"47","xml":"ABCmajorvalue.."}
Example of XML syntax for changing application parameters in post-deployment
step:{"releaseId":"","xml":""}
Example of XML syntax for changing server parameters in post-deployment
step:{"releaseId":"","xml":""}
Response:
{"id":"47","description":"Release updated successfully","result":true}
```

ApplicationApiDto

The application dto contains the application id, application name and description.

Parameters	Type	Description
description	String	Description of the attribute
name	String	Name of the attribute
id	Long	ID of the attribute

ArtifactsApiDto

The Dto contains the available data on an artifact

Parameter	Type	Description
allowArtifactModifications	Boolean	True - a different file can be deployed on the same release settings. False - Only one file is allowed to be executed in the process.
description	String	The description of the artifact (entered by the user).

Parameter	Type	Description
getter type	String	Artifact retrieval method {Agent Local File, Remote File (renamed 'FTP' in 4.2 and renamed back to 'Remote file' in 4.7), SVN, HTTP, SSH, TFS, Repository, FTP}
properties	Map	The properties of the getter type used.
server	String	Represents server cluster and is either single server id or server group id. Is determined by serverGroup property. if serverGroup==true then serverId is ServerGroup id, i.e. Category id. In this case the actual file getter will be selected out of the server group.
serverGroup	Boolean	Indicates whether the above server (artifact getter) is a single server or a server group
shouldUploadNow	Boolean	Should the artifact be stored in the repository now
storeInRepository	Boolean	Should the artifact be stored in the repository on first use
application	String	Application name
applicationId	long	Application ID
artifact	String	Artifact name
artifactId	long	Artifact ID
version	String	Artifact version

ArtifactBasicApiDto

Specify an artifact in one of the following ways::

- Artifact ID {artifactId}
- Application ID {applicationId}, Artifact name {artifact} and Artifact version {artifactVersion}
- Application name {application}, Artifact name {artifact} and Artifact version {artifactVersion}

Parameter	Type	Description	Required
application	String	Application name	true*

Parameter	Type	Description	Required
applicationId	Long	Application ID	true*
artifact	String	Artifact name	true*
artifactId	Long	Artifact ID	true*
version	String	Artifact version	true*

ArtifactNameDtoList

Parameters	Type	Description
List	List	
List	List	

ArtifactPackageApiDto

Contains the ID or path of an artifact-package

Parameter	Type	Description
application	String	Application name
artifactPackage	String	Artifact package name
artifactPackageId	Long	Artifact package ID

ArtifactPackageUploadDto

Contains the xml of the Artifact Package.

Name	Type	Description	Required
applicationId	Long	The id of application to create the package	true
xml	String	XML	true

ArtifactStatusApiDto

Retrieves the status of the artifact data. When the request fails, the data will contain information about the failure.

Parameter	Type	Description
description	String	Description of the response/Artifact
id	Long	Success - Artifact ID
result	Boolean	Result of the request
status	String	Success - Status of the artifact

ArtifactVersionApiDto

Contains the location of a specific artifact version

Parameter	Type	Description
artifactId	Long	Artifact ID
version	String	Artifact version
application	String	Application name
artifactDefinition	String	Artifact definition name
artifactDefinitionId	Long	Artifact definition ID
artifactType	String	Artifact type name

ArtifactVersionsApiDto

Contains the location of artifact version(s)

Parameter	Type	Description
application	String	Application name
artifactDeiniton	String	Artifact definition name
artifactDefinitionID	Long	Artifact definition ID
artifactType	String	Artifact type name

CreateArtifactApiDto

The Dto contains all the available data required to create an artifact

* Specify an application in one of the mandatory ways:

- Application name {application}
- Application ID {applicationId}

Parameters	Type	Description	Required
allowArtifactModifications	Boolean	True - a different file may be deployed on the same release settings. False - Only one file allowed to execute in the process.	false
application	String	Application name	true*
applicationId	Long	Application ID	true*
artifact	String	Artifact name	true
description	String	The description of the artifact (supplied by the user).	false
getterType	String	Artifact retrieval method {Agent Local File, Remote File (renamed 'FTP' in 4.2 and renamed back to 'Remote file' in 4.7), SVN, HTTP, SSH, TFS, Repository, FTP}	true
properties	Map	The properties of the getter type used	true
server	String	Represents server cluster and is either single server id or server group id. This is determined by serverGroup property. if serverGroup==true then serverId is ServerGroup id, i.e. Category id. In this case the actual file getter will be selected out of the server group.	true
serverGroup	Boolean	Indicates whether the above server (artifact getter) is single server or a server group	true
shouldUploadNow	Boolean	should the artifact be stored in the repository now	false
storeInRepository	Boolean	should the artifact be stored in the repository on first use	false
version	String	Artifact version	false

CreateArtifactForDeploymentPlanDto

Contains the xml of the Artifact Package and the deployment plan. Supply the deployment plan in the following ways:

- Deployment plan ID {DeploymentPlanId
- Deployment plan name {DeploymentPlan}.

In the case additional build version {build} and project {project} names are required, use the following ways to supply the application:

- Application ID {applicationId}
- Application name {application}

Name	Type	Description	Required
applicationId	Long	The id of application to create the package at	true
xml	String	XML	true
application	String	Application name	true**
build	String	Build name	true*
deploymentPlan	String	Deployment plan name	true*
deploymentPlanId	Long	Deployment plan id	true*
project	String	Project name	true*

CreateArtifactPackageApiDto

Contains the name, description and application of artifact package

Parameter	Type	Description
application	String	Application name
applicationId	Long	Application ID
description	String	Artifact-package description
name	String	Artifact-package name
description	String	Description of attribute
name	String	Name of the attribute
id	Long	ID of the attribute

CreateArtifactVersionApiDto

This Dto contains all the available data needed to create an artifact-version.

Parameter	Type	Description
applicationId	Long	
artifactDefinition	String	Artifact-definition name
artifactDefinitionId	Long	The ID of Artifact-definition
artifactType	String	Artifact-type name
allowArtifactModifications	Boolean	True - a different file can be deployed on the same release settings. False - Only one file is allowed to be executed in the process.
applicationId	Long	Application ID
description	String	The description of the artifact (enter by the user).
getterType	String	Artifact retrieval method {Agent Local File, Remote File (renamed 'FTP' in 4.2 and renamed back to 'Remote file' in 4.7), SVN, HTTP, SSH, TFS, Repository, FTP}
properties	Map	The properties of the getter type used.
server	String	Represents server cluster and is either single server id or server group id. This is determined by serverGroup property. if serverGroup==true then serverId is ServerGroup id, i.e. Category id. In this case the actual file getter will be selected out of the server group.
serverGroup	Boolean	Indicates whether the above server (artifact getter) is single server or a server group
shouldUploadNow	Boolean	Should the artifact be stored in the repository now
storeInrepository	Boolean	Should the artifact be stored in the repository on first use
version	String	Artifact version

CreateReleaseApiDto

The dto contains the data required to create a release from a template - individualize a template, define release entities, and validate the steps of the environment before the run.

* Specify a template in one of the following ways:

- Template ID {templateId}
- Template Name {template} and Application Name {application}

Parameters	Type	Description	Required
description	String	Release description	false
doStepsValidation	Boolean	True - In case one step (or more) does not mach the specified environment, the operation will fail. False - All steps that wrap processes that are not assigned to the release environment, will be removed from the template automatically.	false
environment	String	Environment name	true
release	String	Release name	true
releaseType	String	Release type {Minor,Major,Emergency. Default is Minor}	false
version	String	Release version. Can be null	false
application	String	Application name	true-
template	String	Template name	true*
templateId	Long	Template ID	true*

DeploymentApiDto

The required arguments in the JSON object is the deployment plan id, deployment, and application names. The environment is supplied in the following ways:

- Environment Names list {environmets}
- Environment Ids list {environmentIds}

Supply the deployment plan in the following ways:

- Deployment plan ID {DeploymentPlanId}</br>
- Deployment plan name {DeploymentPlan}.

Additional build version {build} and project {project} names are required. The application could be supplied by name or id.

Name	Type	Description	Required
applicationId	Long	Application id	true***
deployment	String	Deployment name	true
deploymentDescription	String	Deployment description	false
environmentIds	List	List of the Environment ids the deployment runs on	true*
environments	List	List of the Environment names the deployment runs on	true*
stageToPerform	string	Execute the stage after a deployment is created. The stages that precede will execute. {None, Initialization, Validation, Approval-Gate, Distribute-Execution-Server, Distribute-Agent, Deployment, Post-Deployment} When None is selected, the deployment is only created. Default - will run all the stages.	false
application	String	Application name	true**
build	String	Build name	true*
deploymentPlan	String	Deployment plan name	true*
deploymentPlanId	Long	Deployment plan id	true*
project	String	Project name	true*

DeploymentResponseApiDto

The dto contains the values of a deployment.

Parameter	Type	Dscription	Required
envId	Long	Environment id of the of the deployment	false
envName	String	Environment name of the deployment	false
id	Long	ID of entity	false
description	String	Description of the operation	false
result	Boolean	Result of the requsest	false

DeploymentPlanApiDto

The required arguments in the JSON object are the deployment plan, build, project, Deployment template, application, and template category names. Specify a template category in the following ways:

1. Template Category ID {templateCategoryId}
2. Template Category Name {templateCategory}

The deployment name is required when activating the "run-deployment-plan" REST call. The environment is provided in the following ways:

1. Environment Names list {environmets}
2. Environment Ids list {environmentIds}

The application is supplied by name or id.

Name	Type	Description	Required
application	String	Application name	true***
applicationId	Long	Application id	true***
artifactPackage	String	Artifact package name. Assign the package name, and the package will be assigned to the deployment plan.	false
artifactPackageAsXML	String	Contains the xml of Artifact Package	false

Name	Type	Description	Required
build	String	Build version name	true
deployment	String	Deployment name	true**
deploymentDescription	String	Deployment description	false
deploymentPlan	String	Deployment plan name	true
deploymentPlanDescription	String	Deployment plan description	false
deploymentStageToPerform	String	Execute the stage after deployment has been created. All The stages preceding will be executed. {None, Initialization, Validation, Approval-Gate, Distribute-Execution-Server, Distribute-Agent, Deployment, Post-Deployment} **Note:** If None has been selected, only the deployment is created. Default - run all the stages.	false
deploymentTemplate	String	Deployment template name	true
environmentIds	List	List of environment ids the deployment runs on	true**
environments	List	List of environment names the deployment runs on	true**
manifest	String	Manifest xml. If supplied, the parameters' values (in release scope) is loaded to the deployment plan.	false
project	String	Project name. A project is created if none exist	true*
templateCategory	String	Template category name	true*
templateCategoryId	Long	Template category id	true*

DeploymentPlanResponseApiDto

The dto contains all the available values of a deployment plan.

Name	Type	Description	Required
application	String	Application name	false
artifactPackage	String	Artifact package name	false

Name	Type	Description	Required
build	String	Build version name	false
deploymentPlan	String	Deployment plan name	false
deploymentPlanDescription	String	Deployment plan description	false
deploymentPlanId	Long	Deployment plan id	false
deploymentResults	DeploymentResponseApiDto[]	Deployments created and/or run from the deployment plan	false
deploymentTemplate	String	Deployment template name	false
deploymentsStatus	DeploymentStatusApiDto[]	The Deployments status	false
project	String	Project name	false
result	Boolean	Result of the request	false
templateCategory	String	Template category name	false

EnvironmentApiDto

The environment dto. The fields here are the environment name, environment id, application name, application id, and description.

Parameter	Type	Description
applicationId	Long	Application ID
applicationName	String	Application Name
description	String	Description of the attribute
name	String	Name of the attribute
id	Long	ID of the attribute

EnvironmentParameterApiDto

This dto contains the basic fields that describe a parameter for a specific environment.

Parameter	Type	Description
application	String	Application Name
applicationId	Long	Application ID
environment	String	Environment Name
environmentId	Long	Environment ID
parameterPath	String	Parameter Path

FullEnvironmentParameterApiDto

This dto contains all the fields that are needed to update the value of an environment parameter

Parameter	Type	Description
application	String	Application name
applicationId	Long	Application ID
environment	String	Environment Name
environmentId	Long	Environment ID
parameterPath	String	Parameter Path
serverType	Map	A Map of Server Types and correlated values
arrayValue	List	Value of Array Parameter
loopValue	List	Value of Loop Folder Parameters
simpleValue	String	Value of Simple Parameter

LoadManifestApiDto

Contains the manifest xml and the deployment plan details. Supply the deployment plan in one of the following ways:

- Deployment plan ID {DeploymentPlanId}
- Deployment plan name {DeploymentPlan}.

In this case additional build version {build} and project {project} names are required. Supply the application in one of the following ways:

- Application ID {applicationId}
- Application name {application}

Name	Type	Description	Required
applicationId	Long	Application id	true**
manifest	String	Manifest XML file	true
application	String	Application name	true**
build	String	Build name	true*
deploymentPlan	String	Deployment plan name	true*
deploymentPlanId	Long	Deployment plan id	true*
project	String	Project name	true*

ProjectApiDto

This dto contains the name, description and application of a project.

To create a project, specify the application id {applicationId}, the name of the project {name}, and optionally the description {description}. The id {id} of the created project is returned.

Name	Type	Description	Required
applicationId	Long	Application Id to which this project is associated with.	true
archived	Boolean	Status of the project. Non-active/deleted projects are marked archived.	false
description	String	Project description	false

Name	Type	Description	Required
id	Long	Project Id	false
name	String	Project name	true

ReleaseApiDto

The Dto contains the available data on a release.

Parameter	Type	Description
applicationId	Long	Application ID
applicationName	String	Application Name
endTime	String	End time
environmentId	Long	Environment ID
environmentName	String	Environment name
releaseStatus	String	Release Status - since V2
stage	String	Current stage of the release
stageState	String	Current state of the stage of the release
startTime	String	Start time
status	String	Release status - V1 only
templateId	Long	template ID
templatename	String	Template name
version	String	Release version
description	String	Description of the attribute
name	String	Name of the attribute
id	Long	ID of the attribute

ReleaseBasicApiDto

The dto contains the basic fields for specifying a release.

Specify a release in one of the following way:

- Use Release ID {releaseId}

Use Release name {release}, Release version {releaseVersion}, Application name {application} and Environment name {environment}

Parameters	Type	Description	Required
application	String	Application Name	true *
environment	String	Environment Name	true *
release	String	Release Name	true *
releaseId	Long	Release ID	true *
version	String	Release version Can be null	true *

ReleaseStatusApiDto

The Dto is divided into two sections: Success and Failure, not all fields appear in both sections

- Success - all the fields contain data linked to the release.
- Failure - all the fields contain data regarding the reason of the failure.

Parameter	Type	Description
description	String	Success - general status and progress of the release, Failure - reason of failure
id	Long	Success - Release ID
releaseStatus	String	Success - V2: Release Status {Active/Succeeded/Failed/Canceled}
result	Boolean	Result of the request
stage	String	Success - V2: Current stage of the release {Initialization/Approval-Gate/Deployment/Post-Deployment}

Parameter	Type	Description
stageState	String	Success - V2: Status of the current stage: {Pending/Running/Paused/Running-With-Errors/Succeeded/Failed/Canceled}
status	String	Success - V1: Current state of the release: {Active/Finished/Failed/Canceled}

ResponseData

The general Dto that is returned to the user. The data in the dto is concurrent with the success or the failure of the request.

Parameter	Type	Description
description	String	Description of the response/Attribute
id	Long	Success - ID of the attribute
result	Boolean	Result of the request

ResponseDataApiDto

This is the general Dto that is returned to the user. The data in this dto is concurrent with the success or the failure of the request.

Parameter	Type	Description
description	String	Description of the Response/Attribute
result	Boolean	Result of the request

ResponseEnvironmentParameterApiDto

This dto contains the fields that describe the value of parameter.

Note: This rest call is for version 3 only

Parameter	Type	Description
description	String	Description of the Response/Attribute

Parameter	Type	Description
result	Boolean	Result of the request
parameterPath	String	Parameter Path
serverTypes	Map	A Map of Server Types and correlated values
arrayValue	List	Value of an Array Parameter
loopValue	List	Value of an Loop Folder Parameter
simpleValue	String	Value of a Simple Parameter

RunReleaseApiDto

The dto contains the data required to run a release - individualize a release, define the timeout, and to run asynchronously or not.

Specify a release in one of the following ways:

- Use Release ID {releaseId}
- Use Release name {release}, Release version {releaseVersion}, Application name {application} and Environment name {environment}

Parameter	Type	Description	Required
asynchronously	Boolean	True/false. When true, http call returns after the run release command is triggered, and not after the release run finished. Default is false, http call returns after release	true *
timeout	Long	Timeout in second. When a release run exceeds the timeout, the release stops automatically	true *
application	String	Application Name	true *
environment	String	Environment Name	true *
release	String	Release Name	true *
releaseId	Long	Release ID	true *
vVersion	String	Release version. Can be null	false *

RunTemplateApiDto

The dto contains the data required to run a template which includes: create release, update release, and run release.

Specify a template in one of the following ways:

- Use Template ID {templateId}
- Use Template Name {template} and Application Name {application}

Parameter	Type	Description	Required
description	String	Release description	false
doStepsValidation	Boolean	True - In case one step, or more, does not match the specified environment, the operation fails. False - Wrap process steps not assigned to the release environment, are removed from the template automatically.	false
environment	String	Environment name	true
properties	Map	{map of {key,value}}	false
release	String	Release name	true
release type	String	Release type {Minor,Major,Emergency. Default is Minor}	false
version	String	Release version. Can be null	false
application	String	Application name	true *
template	String	Template name	true *
templateId	Long	Template ID	true *

ScheduleReleaseApiDto

The dto contains the data required to schedule a release - specify a release and define the scheduled date, time and duration.

Specify a release in one of the following ways:

- Use Release ID {releaseId}
- Use Release name {release}, Release version {releaseVersion}, Application name {application} and Environment name {environment}

Parameter	Type	Description	Required
estimatedDurationMinutes	Long	The duration in minutes in the format: mmmm	true
scheduleddate	String	The date in the format: dd/mm/yy	true
scheduledTime	String	The time in the format: HH:mm	true
application	String	Application name	true *
environment	String	Environment name	true *
release	String	Release name	true *
releaseId	Long	Release ID	true *
version	String	Release version. Can be null	false *

StepApiDto

The dto contains just the step ID.

Parameter	Type	Description	Required
stepID	Long	Step ID	true

TemplateApiDto

The Dto contains the available data for a template

Parameter	Type	Description
applicationId	Long	Application ID

Parameter	Type	Description
applicationName	String	Application Name
description	String	Description of the template
status	String	Template status
description	String	Description of the attribute
name	String	Name of the attribute
id	Long	ID of the attribute

TemplateBasicApiDto

The dto contains the basic fields for specifying a template.

Specify a template in one of the following ways:

- Use Template ID {templateId}
- Use Template Name {template} and Application Name {application}

Parameter	Type	Description	Required
application	String	Application name	true *
template	String	Template name	true *
templateId	Long	Template ID	true *

UpdateReleaseApiDto

The dto contains the data required to update a release. Specify a release and the xml that describes the updates.

Specify a release in one of the following ways:

- Use Release ID {releaseId}
- Use Release name {release}, Release version {releaseVersion}, Application name {application} and Environment name {environment}

Parameter	Type	Description	Required
xml	String	Xml describes the required updates	true
application	String	Application name	true *
environment	String	Environment name	true *
release	String	Release name	true *
releaseId	Long	Release ID	true *
version	String	Release version. Can be null	false *