CA Workload Automation
Agent for Web Services

CLI User Guide
r11.3.1
This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the “Documentation”) is for your informational purposes only and is subject to change or withdrawal by CA at any time.

This Documentation may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This Documentation is confidential and proprietary information of CA and may not be disclosed by you or used for any purpose other than as may be permitted in (i) a separate agreement between you and CA governing your use of the CA software to which the Documentation relates; or (ii) a separate confidentiality agreement between you and CA.

Notwithstanding the foregoing, if you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION “AS IS” WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with “Restricted Rights.” Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Copyright © 2013 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.
This document references the following CA Technologies products:

- CA Process Automation
- CA Workload Automation AE
- CA Workload Automation Agent for Application Services (CA WA Agent for Application Services)
- CA Workload Automation Agent for Databases (CA WA Agent for Databases)
- CA Workload Automation Agent for i5/OS (CA WA Agent for i5/OS)
- CA Workload Automation Agent for Linux (CA WA Agent for Linux)
- CA Workload Automation Agent for Micro Focus (CA WA Agent for Micro Focus)
- CA Workload Automation Agent for Microsoft SQL Server (CA WA Agent for Microsoft SQL Server)
- CA Workload Automation Agent for Oracle E-Business Suite (CA WA Agent for Oracle E-Business Suite)
- CA Workload Automation Agent for PeopleSoft (CA WA Agent for PeopleSoft)
- CA Workload Automation Agent for Remote Execution (CA WA Agent for Remote Execution)
- CA Workload Automation Agent for SAP (CA WA Agent for SAP)
- CA Workload Automation Agent for UNIX (CA WA Agent for UNIX)
- CA Workload Automation Agent for Web Services (CA WA Agent for Web Services)
- CA Workload Automation Agent for Windows (CA WA Agent for Windows)
- CA Workload Automation CA 7 Edition
- CA Workload Automation DE
- CA Workload Automation Desktop Client (CA WA Desktop Client)
- CA Workload Automation ESP Edition
- CA Workload Control Center
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: Overview .......................................................... 7
About This Guide ........................................................................................................................................................... 7
Using the Command Line Interface ......................................................................................................................... 8

Chapter 2: Invoking Document/Literal Web Services ................................................................. 9
Web Services Command Line Interface ......................................................................................................................... 9
Invoking the Web Services CLI ..................................................................................................................................... 10
ws_cli Script—Invoke Document/Literal Web Services .............................................................................................. 12
  Example: Invoke Document/Literal Web Services ................................................................................................... 19
  Example: Pass Parameters to a Web Service ............................................................................................................... 20
  Example: Specify WS-Security Options ....................................................................................................................... 21
ws_ingen Script—Generate Input Template for a Web Service Operation ....................................................................... 27

Chapter 3: Integrating with CA Process Automation ................................................................. 31
Process Automation Command Line Interface ............................................................................................................. 31
Invoking the Process Automation CLI .......................................................................................................................... 32
itpam_call Script—Integrate with CA Process Automation ............................................................................................. 33
  Example: Execute a Process Automation Process and Track it to Completion ........................................................... 37
  Example: Abort a Previously Started Process ........................................................................................................... 37
  Example: Display the Entire Process Log .................................................................................................................... 38
  Example: Display Selected Information from the Process Log Using Filters ............................................................. 39
  Example: Retrieve the Process State .......................................................................................................................... 39
  Example: Submit a Start Request Form and Track it to Completion ............................................................................. 40
  Example: Retrieve the Start Request Form Status ....................................................................................................... 40
  Example: List All Start Request Forms ....................................................................................................................... 41
  Example: Display a Start Request Form ........................................................................................................................ 41
  Example: Retrieve the Status of the Process Automation Server .................................................................................. 42

Chapter 4: Running the CLI Using Your Scheduling Manager .................................................. 43
Integration with CA Workload Automation DE ........................................................................................................... 43
  Example: Invoke a Web Service to Get Movie Information Using a UNIX Job .......................................................... 44
  Example: Invoke a Web Service to Get Weather Information Using a Windows Job ................................................... 45
  Example: Invoke a CA Workload Automation DE Web Service .................................................................................. 45
  Example: Execute a Process Automation Process Using a UNIX Job ......................................................................... 49
  Example: Submit a Start Request Form Using a Windows Job ..................................................................................... 50
Integration with CA Workload Automation AE ...........................................................................................................50
  Example: Invoke a Web Service to Get Movie Information Using a Command Job............................................51
  Example: Invoke a Web Service to Get Weather Information Using a Command Job......................................53
  Example: Execute a Process Automation Process Using a Command Job ..........................................................54
  Example: Submit a Start Request Form Using a Command Job ..............................................................................55
Integration with CA Workload Automation ESP Edition ..........................................................................................56
  Example: Invoke a Web Service to Get Movie Information Using a UNIX Job ....................................................57
  Example: Invoke a Web Service to Get Weather Information Using a Windows Job ......................................58
  Example: Execute a Process Automation Process Using a UNIX Job ................................................................59
  Example: Submit a Start Request Form Using a Windows Job ..............................................................................60
Integration with CA Workload Automation CA 7 Edition .........................................................................................60
  Example: Invoke a Web Service to Get Movie Information Using a UNIX Job ....................................................61
  Example: Invoke a Web Service to Get Weather Information Using a Windows Job ......................................62
  Example: Execute a Process Automation Process Using a UNIX Job ...............................................................63
  Example: Submit a Start Request Form Using a Windows Job ..............................................................................63
Chapter 1: Overview

This section provides an overview of the Command Line Interface (CLI).

This section contains the following topics:

About This Guide (see page 7)
Using the Command Line Interface (see page 8)

About This Guide

This guide describes how to use the Command Line Interface (CLI) to invoke document/literal web services and integrate with CA Process Automation. The guide is intended for scheduling developers and testers.

The CLI lets you invoke agent workload that your scheduling manager does not currently support. For example, if your scheduling manager does not support Web Service Document/Literal jobs, you can use the CLI to run this workload as Windows or UNIX (command) jobs. In a future release of your scheduling manager, a Web Service Document/Literal job type will be added to support this workload. At that time, you can upgrade your scheduling manager to take advantage of the new job type or continue to use the CLI.

Note: For information about running agent workload using the supported job types, see the documentation for your scheduling manager.
Using the Command Line Interface

The CLI is a standalone utility that is shipped with the agent as a shell script (UNIX) or batch script (Windows). To use the CLI, you define a UNIX or Windows (command) job in your scheduling manager. In the job definition, you specify the name of the shell or batch script and pass the necessary arguments.

The CLI writes all of the output data to standard output (stdout). When run as a UNIX or Windows (command) job, the agent captures the output into the job's spool file.

For example, to invoke a document/literal web service, you pass the WSDL URL, service name, port name, operation, target endpoint address URL, and other arguments. When the job is submitted, the shell or batch script runs on the agent and invokes the web service residing on the web server. The response is stored in the job's spool file.

To run a Process Automation process, you pass the user, password, process name, target endpoint address URL, and other arguments. When the job is submitted, the shell or batch script runs on the agent and executes the process residing on the CA Process Automation server. The response is stored in the job's spool file.

The following diagram shows how to use the CLI to invoke a web service or run a CA Process Automation process:
Chapter 2: Invoking Document/Literal Web Services

The term web service describes a standardized method for exchanging data between applications and systems. Web services use XML to code and decode the data and Simple Object Access Protocol (SOAP) to transfer it.

Web Service Description Language (WSDL) is an XML-based language that describes a web service and how to access it. A WSDL document specifies the location of the service and the operations the service exposes.

The CA WA Agent for Web Services supports the invocation of document/literal style of web services. Using the agent, you can call an operation within a web service and pass parameters to the operation using document/literal style binding. The parameters represent a flattened view of the XML document the agent constructs.

This section contains the following topics:
- Web Services Command Line Interface (see page 9)
- Invoking the Web Services CLI (see page 10)
- `ws_cli` Script—Invoke Document/Literal Web Services (see page 12)
- `ws_ingen` Script—Generate Input Template for a Web Service Operation (see page 27)

Web Services Command Line Interface

The Web Services CLI is a standalone utility that is shipped with the agent. You can use the CLI to invoke Web Service Document/Literal jobs that your scheduling manager does not currently support.

On UNIX, the Web Services CLI is a shell script. You can run the shell script from the command line or you can schedule it to run using your scheduling manager, similar to other shell scripts. For example, you can define a UNIX (command) job to invoke document/literal web services using the CLI.

On Windows, the Web Services CLI is a batch script. You can run the batch script from the command line or you can schedule it to run using your scheduling manager, similar to other batch scripts. For example, you can define a Windows (command) job to invoke document/literal web services using the CLI.
Invoking the Web Services CLI

You can invoke the Web Services CLI to invoke document/literal web services using the agent.

The Web Services CLI (ws_cli) is located in the following directory:

- On UNIX:
  
  ```
  install_dir/wrappers
  ```

- On Windows:
  
  ```
  install_dir\wrappers
  ```

**install_dir**

Specifies the agent installation directory.

To invoke the CLI, specify the following command:

```shell
wrappers/ws_cli --argument ...
```

**--argument**

Specifies a CLI argument. Prefix each argument with two dashes (--) . The arguments fall under the following categories:

**Mandatory arguments**

Specifies mandatory arguments including the WSDL URL, service name, port name, operation, and the target endpoint address URL.

**Proxy settings**

Specifies optional settings for communicating through a proxy server. The settings include the proxy host, proxy port, proxy user, proxy password, your workstation, and user’s domain.

**Target authentication settings**

Specifies optional settings for authenticating against the target web server that hosts the web service. The settings include the host, port, user, password, your workstation, user’s domain, and authentication order.

**Parameters**

Specifies optional parameters that the agent passes to the web service. Each parameter can be a scalar string value, an array of values, or a serialized Java object that another job produced.
WSS parameters

Specifies SOAP-level WS-Security (WSS) options that the agent uses to communicate with the web server.

- Add a user name token and password to each message that is validated in the SOAP layer of the web service.
- Add a timestamp, with a time-to-live parameter, to each message to prevent message hijacking and reuse.
- Add a digital signature to the message, using a certificate specified by the keystore parameters.
- Encrypt messages using an X509 certificate or a key specified in the job definition.

Keystore parameters

Specifies optional WSS keystore parameters that are only used if the WSS parameters are set. The settings include the path to the keystore, client keystore alias, and keystore password.

Truststore parameters

Specifies truststore parameters that are required if HTTPS is used to communicate with the web service. The settings include the path to the cacerts file on the agent, including the filename cacerts, and the encrypted password.

Success criteria

Defines optional success criteria for the job.

Call switches

Controls how the call is made. You can use --verbose to display the HTTP traffic for the call or --help to display the online help.

Note: By default, install_dir is the working directory. Do not change the working directory when invoking the Web Services CLI.

Example: Display the Online Help

To display the online help, enter the following command:

wrappers/ws_cli --help
ws_cli Script—Invoke Document/Literal Web Services

The ws_cli script lets you invoke document/literal web services.

The script has the following format:

```
wrappers/ws_cli --wsdl=wsdl_url --svcname=service --portname=portname
   --operation=operation --endpoint=endpoint_URL
   [--proxy[proxy_setting]=proxy_value ...]
   [--target_auth[target_authentication_setting]=setting_value ...]
   [--param[parameter_name]=parameter_value ...]
   [--wssparm[wss_parameter_name]=wss_parameter_value ...]
   [--keystore[keystore_argument]=keystore_argument_value ...]
   [--truststore[truststore_argument]=truststore_argument_value ...]
   [--criteria[success_criteria_name]=success_criteria_value ...]
   [--verbose] [--help]
```

```
--wsdl=wsdl_url
```

Specifies the URL to the Web Service Description Language (WSDL) of the web service to invoke.

```
--svcname=service
```

Specifies the web service name within the target namespace.

```
--portname=portname
```

Specifies the WSDL portname within the target namespace. A WSDL port describes the operations exposed by a web service and defines the connection point to the web service.

```
--operation=operation
```

Specifies the operation to be invoked.

```
--endpoint=endpoint_URL
```

Specifies the target endpoint address URL. In a published WSDL file, the URL defining the target endpoint address is found in the location attribute of the port's soap:address element.

```
--proxy[proxy_setting]=proxy_value ...
```

(Optional) Specifies optional settings for communicating through a proxy server. Specify one entry per setting. Options for proxy_setting are as follows:

```
host
```

Specifies the host name of the proxy server.

```
port
```

Specifies the port on the server of the proxy.
user

Specifies the user name for proxy authentication.

password

Specifies the proxy server password.

Note: The password must be encrypted. To encrypt a password, use the Password utility that is provided with the agent. For more information about the Password utility, see the CA Workload Automation Agent for UNIX, Linux, or Windows Implementation Guide.

Important! We recommend that you secure and restrict access to any job definitions that contain encrypted passwords.

workstation

Specifies the origin name for proxy authentication.

domain

Specifies the domain for proxy authentication.

--target_auth[|target_authentication_setting|]=setting_value ...

(Optional) Specifies optional settings for authenticating against the target web server that hosts the web service. Specify one entry per setting. Options for target_authentication_setting are as follows:

host

Specifies the host name of the target web server.

port

Specifies the port of the target web server.

user

Specifies the user name for connection authentication.

password

Specifies the password for the client authentication user.

Note: The password must be encrypted. To encrypt a password, use the Password utility that is provided with the agent. For more information about the Password utility, see the CA Workload Automation Agent for UNIX, Linux, or Windows Implementation Guide.

Important! We recommend that you secure and restrict access to any job definitions that contain encrypted passwords.

workstation

Specifies the origin host name for NTLM connection authentication.
domain

Specifies the domain for NTLM connection authentication.

order

Specifies a list of protocols used by a web server for authentication. Specify any of the following protocols in any order: BASIC, DIGEST, and NTLM. To specify more than one protocol, separate the protocols by commas.

--param[parameter_name]=parameter_value ...

(Optional) Specifies optional parameters that the agent passes to the web service. Options are as follows:

--param[name]="value"

Specifies the parameter name and value.

name

Specifies the name or path of the element.

value

Specifies the scalar string value.

--param[name][]="value,value,..."

Specifies the parameter name and array of values.

name[

Specifies the name or path of the element. When the parameter name contains [i], it indicates an array of values.

value,value,...

Specifies an array of scalar string values. The individual array values are separated by commas and the entire value is enclosed in double quotes.

--param[name[URI]]="URI"

Specifies the parameter name and serialized Java object.

name[URI]

Specifies the name or path of the element. When the parameter name contains [URI], it indicates a serialized Java object.

URI

Specifies the location (URI) of the binary output produced by a payload producing job. A payload producing job is a job that produces binary output that is persisted as a serialized object. The serialized Java object is stored as a file on the agent computer. The URI can be passed as input to a payload consuming job.
Notes:

- To specify an XML hierarchy of the document, use multiple entries of --param.

- In the XML document, these parameters are represented in a flattened view. Consider the following example:

```xml
<tag1>
  <tag2>value</tag2>
<tag1>
```

In this example, the full path to the value is </tag1/tag2>.

- For tags that do not hold any value, the full path is required because the no-value tags could represent arrays of structures and appear in the multiple levels. Consider the following example:

```bash
--param["/tag1"]=""
--param["/tag1/tag2"]="value"
```

- For parameters that contain attributes, the attributes are added to the full path of the parameter separating them with the '@' symbol. Consider the following example:

```xml
<tag1>
  <tag2 key="thetag">value</tag2>
</tag1>
```

In this example, the list of parameters would be as follows:

```bash
--param["/tag1"]=""
--param["/tag1/tag2"]="value"
--param["/tag1/tag2@key"]="thetag"
```

- To help you code the parameters to the web service, the agent includes a utility that generates the input template for a web service operation. The utility is named `ws_ingen` (see page 27) and is included in the same directory as the Web Services CLI.

- For Web Service Document/Literal payload producing jobs, only the portion of the returned data that satisfies the --criteria specification will be stored in the output file for use as payload.
--wssparm[wss_parameter_name]=wss_parameter_value ...

(Optional) Specifies SOAP-level WS-Security (WSS) options that the agent uses to communicate with the web service.

Important! The agent outputs the XML documents that it sends and receives into the spool file. We recommend that you secure and restrict access to the spool files to protect sensitive information.

The WSS options are as follows:

--wssparm[USERTOKEN]="User=user_name, Password=password, Digest=TRUE | FALSE"

Adds a user name and password to the SOAP message that is validated in the server’s SOAP layer.

User=user_name

Specifies the user name that is added to the SOAP message.

Password=password

Specifies the password for the user name that is added to the SOAP message.

Note: The password must be encrypted. To encrypt a password, use the Password utility that is provided with the agent. For more information about the Password utility, see the CA Workload Automation Agent for UNIX, Linux, or Windows Implementation Guide.

Important! We recommend that you secure and restrict access to any job definitions that contain encrypted passwords.

Digest=TRUE | FALSE

Specifies whether you want to send the password in the encrypted form (TRUE) or in plain form (FALSE) to the SOAP request.

Important! We recommend that you set Digest to TRUE. If Digest is set to FALSE, the agent writes the decrypted plain text password into the spool file.

--wssparm[TIMESTAMP]="TTL=time_to_live"

Adds a timestamp to the SOAP message with an optional time-to-live option to prevent reuse of old messages.

TTL=time_to_live

Specifies the time-to-live in seconds.

--wssparm[SIGNATURE]="

Adds a digital signature to the SOAP message, using the certificate specified using the keystore argument.
--wssparm[ENCRYPTION]="ServiceAlias=alias, {KeyType=ISSUER_SERIAL} | {KeyType=EMBEDDED_KEYNAME KeyName=key_name KeyValue=key_value}"

Encrypts the message. The server will decrypt the message in the SOAP layer.

**ServiceAlias=alias**

Specifies the alias in the keystore defined by the keystore argument for the web service certificate.

**KeyType=ISSUER_SERIAL**

Uses X509 certificate for encryption. It is obtained from the keystore specified using the keystore argument.

**KeyType=EMBEDDED_KEYNAME**

Uses the following key specified in the job definition for encryption:

**KeyName=key_name**

Specifies the key name as defined by the web service to be invoked.

**KeyValue=key_value**

Specifies the key in hexadecimal form as defined by the web service to be invoked. The agent sends the key encrypted.

--keystore[keystore_argument]=keystore_argument_value ...

(Optional) Specifies optional WSS keystore parameters. The keystore parameters are only used if the WSS parameters are set. Options for keystore_argument are as follows:

**path**

Specifies the full path to your keystore file that contains the X509 certificate. You can generate your own keystore using the keytool utility provided with the JRE.

**alias**

Specifies the client keystore alias. The alias is for the agent's public or private keys.

**password**

Specifies the keystore password. The password is encrypted in hexadecimal form.

**Note:** The password must be encrypted. To encrypt a password, use the Password utility that is provided with the agent. For more information about the Password utility, see the CA Workload Automation Agent for UNIX, Linux, or Windows Implementation Guide.

**Important!** We recommend that you secure and restrict access to any job definitions that contain encrypted passwords.
--truststore[truststore_argument]=truststore_argument_value ...

(Optional) Specifies truststore parameters that are required if HTTPS is used to communicate with the web service. Options for truststore_argument are as follows:

path

Specifies the full path to the cacerts file on the agent, including the filename cacerts.

Example: /opt/ORACLE_SSL/ESPSystemAgent/cacerts

Note: Import the SSL certificate into the cacerts file on the agent using the keytool utility that is provided with the JRE.

password

Specifies the password for accessing the truststore file. The password is encrypted in hexadecimal form.

Note: The password must be encrypted. To encrypt a password, use the Password utility that is provided with the agent. For more information about the Password utility, see the CA Workload Automation Agent for UNIX, Linux, or Windows Implementation Guide.

Important! We recommend that you secure and restrict access to any job definitions that contain encrypted passwords.

--criteria[success_criteria_name]=success_criteria_value ...

(Optional) Defines optional success criteria for the job. Options for success_criteria_name are as follows:

xpath

(Optional) Checks the xpath expression that is used to evaluate a return string. In the xpath expression, use the namespace prefix specified in the WSDL. If the WSDL contains multiple prefixes for the same value, the first occurrence is used.

filter

Defines a regular expression that is used to evaluate a return string. If the return string matches the regular expression, the job completes successfully. Otherwise, the job fails.

Note: If the xpath expression is not present, the filter expression applies to the entire response returned by the call. The response is transformed into a formatted XML document before being checked against the filter. The returned XML is formatted to be easier to read with a four-space indent for nested tags. Specifically, the following keys are used for transformation:

“indent” : "yes"

“standalone” : “yes”

"{http://xml.apache.org/xslt}indent-amount" : "4"
--verbose
(Optional) Displays the HTTP traffic for the call.

--help
(Optional) Displays online help.

Example: Invoke Document/Literal Web Services

This example calls a web service that retrieves the version of Apache Axis 2:

```
--svcname=Version --portname=VersionHttpSoap11Endpoint --operation=getVersion
```

The request is as follows:

Request
-------

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
xmlns:ns="http://axisversion.sample">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body/>
</SOAP-ENV:Envelope>
```

The response is as follows:

Response
-------

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns:getVersionResponse xmlns:ns="http://axisversion.sample">
      <ns:return>Hi - the Axis2 version is 1.6.1</ns:return>
    </ns:getVersionResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Execution completed successfully
Example: Pass Parameters to a Web Service

This example calls a web service that returns the maximum of an arbitrary number of integers that are passed to it. In this case, the web service is passed three integers – 1, 4, and 2 – and is expected to return 4.

```
wrappers/wscli --wsdl=http://localhost:8080/axis2/services/TestWS?wsdl
--svcname=TestWS --portname=TestWSHttpSoap11Endpoint --operation=intGetMax
--endpoint=http://localhost:8080/axis2/services/TestWS.TestWSHttpSoap11Endpoint
--param[/intGetMax]="" --param[/intGetMax/ii[3]]=""1", "4", "2"
--criteria[xpath]="/ns:return" --criteria[filter]=".*4.*"
```

The request is as follows:

**Request**
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
xmlns:ns="http://ws.test">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <ns:intGetMax xmlns:ns="http://ws.test">
      <ns:ii>1</ns:ii>
      <ns:ii>4</ns:ii>
      <ns:ii>2</ns:ii>
    </ns:intGetMax>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The response is as follows:

**Response**
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns:intGetMaxResponse xmlns:ns="http://ws.test">
      <ns:return>4</ns:return>
    </ns:intGetMaxResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

The value of <4> matches
Execution completed successfully
Example: Specify WS-Security Options

This example calls a web service that checks the returned XML for the value "Hello World". A timestamp, with a time-to-live parameter of 60 seconds, and a digital signature are added to each SOAP message. The SOAP messages are encrypted using an X509 certificate. The full path to the keystore is /opt/security/crypto/client.jks, the client keystore alias is client, and the encrypted keystore password is 2965DF7DA92F8BD.

Important! We recommend that you secure and restrict access to any job definitions that contain encrypted passwords.

wrappers/ws_cli --wsdl=http://localhost:8080/axis2/services/sample06?wsdl
--svcref=sample06 --portname=sample06HttpSoap11Endpoint --operation=echo
--endpoint=http://localhost:8080/axis2/services/sample06.sample06HttpSoap11Endpoi
nt/ --param[echo][]="" --param[echo/args0][]="Hello World"
--criteria[xpath][]="//ns:return" --criteria[filter][]="Hello World"
--wssparm[TIMESTAMP][]="TTL=60" --wssparm[signature][]=""
--wssparm[ENCRYPTION][]="ServiceAlias=\"service\", KeyType=\"ISSUER_SERIAL\"
--keystore[path]="/opt/security/crypto/client.jks" --keystore[alias]="client"
--keystore[password]="A2965DF7DA92F8BD"

The request is as follows:

Request
--------

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
xmlns:ns="http://sample06.samples.rampart.apache.org"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
  <SOAP-ENV:Header>
      <xenc:EncryptedKey Id="EncKeyId-313741541FD2C15A2913274527741775" xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
        <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
        <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
            <ds:X509IssuerSerial>
              <ds:X509IssuerName>CN=Sample Service,OU=Rampart,O=Apache,L=Colombo,ST=Western,C=LK</ds:X509IssuerName>
              <ds:X509SerialNumber>1261391463</ds:X509SerialNumber>
            </ds:X509IssuerSerial>
          </wsse:SecurityTokenReference>
          </ds:X509Data>
        </ds:X509IssuerSerial>
      </xenc:EncryptedKey>
    </wsse:Security>
  </SOAP-ENV:Header>
</SOAP-ENV:Envelope>
<ds:KeyInfo Id="KeyId-313741541FD2C15A2913274527739872">
  <wsse:SecurityTokenReference
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
    wsu:Id="STRId-313741541FD2C15A2913274527739883" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
    <wsse:Reference
      URI="#CertId-313741541FD2C15A2913274527739801"
      ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"
      xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"/>
  </wsse:SecurityTokenReference>
</ds:KeyInfo>

<wsse:Signature xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="StampId">
  <wsu:Timestamp
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="Timestamp-1">
    <wsu:Created>2012-01-25T00:52:53.976Z</wsu:Created>
    <wsu:Expires>2012-01-25T00:53:53.976Z</wsu:Expires>
  </wsu:Timestamp>
</wsse:Security>
</SOAP-ENV:Header>

<xenc:EncryptedData Id="EncDataId-4" xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
  <xenc:EncryptionMethod
    Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
  <ds:KeyInfo
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <wsse:SecurityTokenReference
      xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"/>
  </ds:KeyInfo>
</xenc:EncryptedData>
The response is as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
<soapenv:Header>
<wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" soapenv:mustUnderstand="1">
<xenc:EncryptedKey Id="EncKeyId-17E7834197DE3F9E4132745277426625">...
<xenc:CipherData>
<xenc:CipherValue>G+fSOSxnL7wZXAMvrMBJxkJUYYSCP1rk6Dn4f0GklrGkK0K80BRaoUC8104f9JYtyV8/hvPdZxym7Gw88BNzS6BF0GAdk:b109tMw4UzKzhcdETH8I20h38F/d0RZPgs17Be2nbaqM1Cl281heA9hdS1a/hcKmzJn19AA</xenc:CipherValue>
<xenc:ReferenceList>
<xenc:DataReference URI="#EncDataId-25"/>
</xenc:ReferenceList>
</xenc:CipherData>
</wsse:Security>
</soapenv:Header>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
ws_cli Script
— Invoke Document/Literal Web Services

Chapter 2: Invoking Document/Literal Web Services

<wsse:BinarySecurityToken
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"
wsu:Id="CertId-17E78834197DE3F9E4132745277425821">MIICUTCCAbgAwIBAgIESy9OZzANBgkqhkiG9w0BAQUFADBtMQswCQYDVQQGEwJMSzEQMA4GA1UECBMHV2VzdGluZyBE...XZK N0Z2kMRAwDgYDVQQH<...>s/00/00/xmldsig# Id="Signature-23">
<ds:Signature xmlns="http://www.w3.org/2000/09/xmldsig#"
Id="Signature-23">
<ds:Reference URI="#id-24"/>
<ds:Transforms/>
<ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
<ds:Transforms/>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<ds:DigestValue>DgbDaA5K3VJjWIoDmggWunWjSLQ=</ds:DigestValue>
<ds:Reference URI="#SigConf-21"/>
<ds:Reference URI="#SigConf-21"/>
<ds:Transforms/>
</ds:Transforms>
<ds:Signature/>

CQ2kgwJn06wz9PJR/JF7yutUFBGVoxiz7A6Eo2qbCdwz1ZfAE/VNTUzkZ2XwqyLwvMrDrWz2e9zxf HINxGtELsGbkne0dQ7TB7ajDQYNPEdL03Cj9/tQGRiHUIEIEeWt/D0ZAa6q4MmpNnvuxGu FK3B5ax+FB5rj3R+0NM=</ds:SignatureValue>
<ds:KeyInfo Id="KeyId-17E78834197DE3F9E4132745277425822">
<wsse:SecurityTokenReference
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
wsu:Id="STRId-17E78834197DE3F9E4132745277425822">
<wsse:Reference URI="#CertId-17E7834197DE3F9E4132745277425821" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"/>
</wsse:SecurityTokenReference>
</ds:Signature>
</ds:KeyInfo>
<wsu:Created>2012-01-25T00:52:54.258Z</wsu:Created>
<wsu:Expires>2012-01-25T00:57:54.258Z</wsu:Expires>
</wsu:Timestamp>
</wsse11:SignatureConfirmation xmlns:wsse11="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" Value="NkmDsEJabW1IAnXMXryYz4WEdcqITVD51bjURMPmjfolBy9r4Ry33rUtYajltWCFvANXitfbQ0vfoFeH4HV/snRFlaq4b2tG9afZTf+Ipy+VW6C05XjiiJ/iWLkds8SrSxfkxUgt0M7TQA7wREl191Kqah39d/7FM=" wsu:Id="SigConf-21"/>
</wsse:Security>
</soapenv:Header>
<soapenv:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="id-24">
<xenc:EncryptedData Id="EncDataId-25" Type="http://www.w3.org/2001/04/xmlenc#Content">
<xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
</wsse:SecurityTokenReference>
<wsse:Reference URI="#EncKeyId-17E7834197DE3F9E4132745277425825"/>
</wsse:SecurityTokenReference>
</ds:KeyInfo>
<xenc:CipherValue>cg9qKQIIZLfoIWpvJpcTMLjl1qCfeV/ThDw1j8i+ayZ+xkzb49Mnl0jqPHWJbkSmw80762nRglHb471NoqGDNN7752AgNAe6kmk3q+OM88S3dYy2FgybkKw/fMhW2K201926yY00CgghBn5dUZGkKqru5rcL2qNaiew/dZV6BlE+T3JqzX6b0GmBs/n5vA61rmsov43wGn500NjScV/76c7ZfdLZuO3S+zV/K0TyuwpQnS0jix6yujzyda6yasPsfn0reoxdIE0jIc/uLSMSNq1v2dvNi0PbRaGYVQ4h6adgXdu2g2Ydw/07wadyqgbE9hKTO0PX/kUdzg/D/f5C9/O2wp67Fa0RVb3QjYFg1LYR7sgj+z07DvDfJE+v38goJDi994545we0czcvRv0w8084nphjyHc514ltds16KqCW5pmAVU+4VA1St0ZGz2N/UD0Fzbd6QMdydP664TC/9QA==</xenc:CipherValue>
</xenc:CipherValue>
</xenc:EncryptedData>
ws_ingen Script—Generate Input Template for a Web Service Operation

The ws_ingen script lets you generate the input template for a web service operation. You can use the input template to help you code the parameters to pass to the web service.

The script has the following format:

```
wrappers/ws_ingen --wsdls=wsdl_url --svcname=service --portname=portname
--operation=operation [--proxy[proxy_setting]=proxy_value ...]
[--help]
```

- `--wsdls=wsdl_url`
  Specifies the URL to the Web Service Description Language (WSDL) of the web service to invoke.

- `--svcname=service`
  Specifies the web service name within the target namespace.

- `--portname=portname`
  Specifies the WSDL portname within the target namespace. A WSDL port describes the operations exposed by a web service and defines the connection point to the web service.

- `--operation=operation`
  Specifies the operation to be invoked.

- `--proxy[proxy_setting]=proxy_value ...`
  (Optional) Specifies optional settings for communicating through a proxy server. Specify one entry per setting. Options for `proxy_setting` are as follows:

  - `host`
    Specifies the host name of the proxy server.

  - `port`
    Specifies the port on the server of the proxy.

  - `user`
    Specifies the user name for proxy authentication.
password

Specifies the encrypted proxy server password.

**Important!** We recommend that you secure and restrict access to any job definitions that contain encrypted passwords.

domain

Specifies the domain for proxy authentication.

--help

(Optional) Displays online help.

**Example: Generate Input Template**

This example generates the input template for the ListAllMyBuckets operation of the AmazonS3 web service. The proxy parameters are supplied to access the WSDL outside the corporate firewall.

```
wrappers/ws_ingen --wsdl=https://s3.amazonaws.com/soap --svcname=AmazonS3
--portname=AmazonS3 --operation=ListAllMyBuckets
--proxy[host]=myproxy.mycompany.com --proxy[port]=80 --proxy[user]=me
--proxy[password]=1267AB731267AB73 --proxy[domain]=mydomain
```

The input template is as follows:

```
Name = /executeStartRequest, Value = <group element, no value>
Name = /executeStartRequest/auth, Value = <group element, no value>

# Optional element. Maximum 1 occurrence
Name = /executeStartRequest/auth/seq, Value = Fixed <auth_true>
Name = /executeStartRequest/auth/user, Value = ?
Name = /executeStartRequest/auth/password, Value = ?
Name = /executeStartRequest/objLocation, Value = <group element, no value>
Name = /executeStartRequest/objLocation/name, Value = ?
Name = /executeStartRequest/objLocation/path, Value = ?

# Optional element. Maximum 1 occurrence
Name = /executeStartRequest/params, Value = <group element, no value>
Name = /executeStartRequest/params/param, Value = ?
Name = /executeStartRequest/params/param@name, Value = ?
```
# Optional element. Maximum 1 occurrence

Name = /executeStartRequest/options, Value = <group element, no value>
Name = /executeStartRequest/options/startDate, Value = ?
Name = /executeStartRequest/options/startTime, Value = ?

# Optional element. Maximum 1 occurrence

Name = /executeStartRequest/options/priority, Value = ?
Chapter 3: Integrating with CA Process Automation

CA WA Agent for Web Services supports integration with CA Process Automation (formerly CA IT Process Automation Manager or CA IT PAM). CA Process Automation is a system that enables companies to deliver IT process automation across their IT organizations. Using the agent, you can launch, monitor, and control a CA Process Automation process. You can also retrieve process logs and Start Request Forms and get the status of the CA Process Automation server.

This section contains the following topics:
- Process Automation Command Line Interface (see page 31)
- Invoking the Process Automation CLI (see page 32)
- itpam_call Script—Integrate with CA Process Automation (see page 33)

Process Automation Command Line Interface

The Process Automation CLI is a standalone utility that is shipped with the agent. You can use the CLI to run Process Automation jobs that your scheduling manager does not currently support.

On UNIX, the Process Automation CLI is a shell script. You can run the shell script from the command line or you can schedule it to run using your scheduling manager, similar to other shell scripts. For example, you can define a UNIX (command) job to execute a Process Automation process or submit a Start Request Form using the CLI.

On Windows, the Process Automation CLI is a batch script. You can run the batch script from the command line or you can schedule it to run using your scheduling manager, similar to other batch scripts. For example, you can define a Windows (command) job to execute a Process Automation process or submit a Start Request Form using the CLI.

Note: By default, the Process Automation CLI does not track the executing process to completion on the CA Process Automation server. Without tracking, the CLI call returns when the process starts. However, for monitoring purposes, you can request that the CLI track the executing process to completion. In this case, the CLI call returns when the process completes. If the CLI request completes successfully, the return code of the script is 0. Otherwise, the request fails and the return code is non-zero to indicate an error condition.
Invoking the Process Automation CLI

You can invoke the Process Automation CLI to integrate the agent with CA Process Automation.

The Process Automation CLI (itpam_call) is located in the following directory:

- On UNIX:
  
  \`install_dir/wrappers\`

- On Windows:
  
  \`install_dir\wrappers\`

\`install_dir\`

Specifies the agent installation directory.

To invoke the CLI, specify the following command:

\`wrappers/itpam_call --argument ...\`

\`--argument\`

Specifies a CLI argument. Prefix each argument with two dashes (\`--\`). The arguments fall under the following categories:

**Mandatory arguments**

Specifies mandatory arguments including the user, password, operation, and target endpoint address URL on the CA Process Automation server.

**Main arguments**

Specifies main arguments such as the name of the process or Start Request Form.

**Optional arguments**

Specifies optional arguments to pass to the process or Start Request Form.

**Operation switches**

Lets you supply switches to the operation. Currently, only \`--track\` is supported. The \`--track\` switch lets you track a process or Start Request Form to completion.

**Call switches**

Controls how the call is made. Currently, only two call switches are supported:

- \`--help\`—Displays online help.
- \`--trace\`—Displays all incoming and outgoing SOAP messages between the CLI client and the Process Automation SOAP endpoint.

**Note:** By default, \`install_dir\` is the working directory. Do not change the working directory when invoking the Process Automation CLI.
Example: Display the Online Help

To display the online help, enter the following command:

wrappers/itpam_call --help

The itpam_call script provides integration with CA Process Automation.

The script has the following format:

wrappers/itpam_call --user=user_name --password=password --operation=operation
[--targets=target_namespace] [--svcname=service_name] [--portname=port_name]
--endpoint_address=endpoint_address --mainarg[argument]=argument_value ...
[--param[param_name]=param_value ...] [--option[option_name]=option_value ...]
[--operation_switch[track]=true|false] [--trace] [--help]

--user=user_name

Specifies the Process Automation user name.

--password=password

Specifies the password for the Process Automation user name.

Important! We recommend that you secure and restrict access to any job definitions that contain encrypted passwords.

--operation=operation

Specifies the operation. Options are as follows:

exec_process

Executes a Process Automation process.

control_process

Controls a Process Automation process.

get_process_logs

Displays a Process Automation process log.

get_process_state

Retrieves the state of a Process Automation process.

exec_start_request

Submits a Start Request Form to execute a Process Automation process.

get_start_request_status

Retrieves the status of a Start Request Form.
get_start_request_forms
Lists all Start Request Forms.

get_start_request_form
Displays a Start Request Form.

check_server_status
Retrieves the status of the Process Automation server.

--targetns=target_namespace
(Optional) Specifies the WSDL target namespace.

Note: Do not specify this value unless instructed by CA Support.

--svcname=service_name
(Optional) Specifies the WSDL service name.

Note: Do not specify this value unless instructed by CA Support.

--portname=port_name
(Optional) Specifies the WSDL port name.

Note: Do not specify this value unless instructed by CA Support.

--endpointaddress=endpoint_address
Specifies the target endpoint address URL on the CA Process Automation server.

--mainarg[argument]=argument_value ...
Specifies main arguments for the operation. Specify one entry per argument. The main arguments are as follows:

- exec_process
  --mainarg[name]=process
  Specifies the fully qualified name of the process.

- control_process
  --mainarg[ProcessID]=process_ID
  Specifies the ID of the process to control.

  --mainarg[action]=suspend|resume|abort
  Indicates whether to suspend, resume, or abort the process.

- get_process_logs
  --mainarg[ProcessID]=process_ID
  Specifies the process ID.
- get_process_state
  --mainarg[ROIID]=process_ID
  Specifies the process ID.

- exec_start_request
  --mainarg[name]=name
  Specifies the name of the Start Request Form.
  --mainarg[path]=path
  Specifies the path of the Start Request Form.
  **Note:** The specified path must end with a / character.

- get_start_request_status
  --mainarg[interactionId]=interaction_ID
  Specifies the Start Request Form interaction ID.

- get_start_request_form
  --mainarg[name]=name
  Specifies the name of the Start Request Form.
  --mainarg[path]=path
  Specifies the path of the Start Request Form.
  **Note:** The specified path must end with a / character.

**Note:** The get_start_request_forms and check_server_status operations do not take any main arguments.

--param[param_name]=param_value ...
(Optional) Specifies parameters to pass to the executing process or Start Request Form. Applies to the exec_process and exec_start_request operations. Specify one entry per parameter.

--option[option_name]=option_value ...
(Optional) Specifies options for the get_process_logs operation. Options for option_name are as follows:

--level
  Specifies the filter level. Options are as follows:
  - normal
  - notice
  - warning
  - error
--category
    Specifies the filter category. Options are as follows:
    - process
    - operator
    - handler
    - response
    - other
    - custom

--operation_switch[track]=true|false
    (Optional) Indicates whether to track the executing process or Start Request Form to completion. Applies to the exec_process and exec_start_request operations.

    true
    Tracks the process or Start Request Form to completion. The CLI call returns when the process or Start Request Form completes, whether successfully or not.

    Note: The agent tracks the process or Start Request Form by polling every 10 seconds. You cannot change the polling interval.

    false
    Does not track the process or Start Request Form to completion. This is the default.

    Default: false

--trace
    (Optional) Turns on tracing. The CLI displays all incoming and outgoing SOAP messages between the CLI client and the Process Automation SOAP endpoint.

--help
    (Optional) Displays online help.
Example: Execute a Process Automation Process and Track it to Completion

This example executes a Process Automation process and tracks it to completion. The main argument for name of the process, as well as an optional parameter named strName, are passed. To track the process to completion, the track switch is set to true. If the CLI request completes successfully, the return code of the script is 0. Otherwise, the request fails and the return code is non-zero to indicate an error condition.


In the following response, the first two lines are printed when the process gets started. The first line indicates the process ID. The second line prints an associated message that is returned by the CA Process Automation server.

ROID: 2793
Message: Document accepted for processing

The third line is printed when the process terminates. In this example, the process terminated successfully.

Process completed

Example: Abort a Previously Started Process

This example aborts a previously started process:


The response is as follows:

Status: abort action for Process Id "1456" has been queued.
Example: Display the Entire Process Log

This example displays the entire process log:

```
wrappers/itpam_call --user=pamadmin --password=pamadmin
--endpointaddress=http://lodlc1003.ca.com:8080/itpam/soap
--operation=get_process_logs --mainarg[processID]=2865
```

The response is similar to the following:

Sun Sep 25 20:27:02 EDT 2011 [Process, Notice] 'Demo_WAAE_1_2865' instance has been created

Sun Sep 25 20:27:02 EDT 2011 [Process, Notice] 'Demo_WAAE_1_2865' is in 'Queued' State


Sun Sep 25 20:27:02 EDT 2011 [Operator, Notice] 'Start_WAAE_Job_1' is enabled following 'Start_1'

Sun Sep 25 20:27:02 EDT 2011 [Operator, Notice] 'Start_1' is 'Completed' on 'Current Server'

Sun Sep 25 20:27:02 EDT 2011 [Operator, Notice] service request sent for 'Start_WAAE_Job_1'

Sun Sep 25 20:27:02 EDT 2011 [Operator, Notice] 'Start_WAAE_Job_1' is 'Running' on 'Current Server'

Sun Sep 25 20:27:02 EDT 2011 [Operator, Notice] 'Start_WAAE_Job_1' is 'Failed'

Sun Sep 25 20:27:02 EDT 2011 [Operator, Notice] 'Email_Alert_1' is enabled following 'Start_WAAE_Job_1'

Sun Sep 25 20:27:02 EDT 2011 [Operator, Notice] service request sent for 'Email_Alert_1'

Sun Sep 25 20:27:02 EDT 2011 [Operator, Notice] 'Email_Alert_1' is 'Running' on 'Current Server'

Sun Sep 25 20:27:02 EDT 2011 [Operator, Notice] 'Email_Alert_1' is 'Completed'

Sun Sep 25 20:27:02 EDT 2011 [Operator, Notice] 'Abnormal_Stop_1' is enabled following 'Email_Alert_1'

Sun Sep 25 20:27:02 EDT 2011 [Operator, Notice] 'Abnormal_Stop_1' is 'Failed' on 'Current Server'

Example: Display Selected Information from the Process Log Using Filters

This example uses filters to retrieve information containing the notice level and the process category. The filters are passed using options.

```
wrappers/itpam_call --user=pamadmin --password=pamadmin
--endpointaddress=http://lodlc1003.ca.com:8080/itpam/soap
--operation=get_process_logs --mainarg[processID]=2865 --option[level]=notice
--option[category]=process
```

The response is similar to the following:

Sun Sep 25 20:27:02 EDT 2011 [Process, Notice] 'Demo_WAAE_1_2865' instance has been created

Sun Sep 25 20:27:02 EDT 2011 [Process, Notice] 'Demo_WAAE_1_2865' is in 'Queued' State


Example: Retrieve the Process State

This example retrieves the state of the process with a process ID of 2865:

```
wrappers/itpam_call --user=pamadmin --password=pamadmin
--endpointaddress=http://lodlc1003.ca.com:8080/itpam/soap
--operation=get_process_state --mainarg[ROID]=2865
```

The following response indicates that the process has failed:

Process state: FAILED
**Example: Submit a Start Request Form and Track it to Completion**

This example submits a Start Request Form and tracks it to completion. The main arguments for name and path of the Start Request Form, as well as an optional parameter named strName, are passed. To track the Start Request Form to completion, the track switch is set to true.

```bash
wrappers/itpam_call --user=pamadmin --password=pamadmin
--endpointaddress=http://lodlc1003.ca.com:8080/itpam/soap
--operation=exec_start_request --mainarg[name]=Demo_WAAE_1_SRF
--mainarg[path]=/CAWAAE/ --param[strName]=CAPATest --operation_switch[track]=true
```

In the following response, the first line indicates the interaction ID. The second line indicates that the Start Request Form has completed successfully.

```
InteractionID:3216
Start Request completed
```

**Example: Retrieve the Start Request Form Status**

This example retrieves the status of the Start Request Form with an interaction ID of 3216:

```bash
wrappers/itpam_call --user=pamadmin --password=pamadmin
--endpointaddress=http://lodlc1003.ca.com:8080/itpam/soap
--operation=get_start_request_status --mainarg[interactionId]=3216
```

The following response indicates that the Start Request Form object has completed successfully:

```
Start Request status: COMPLETED
```
Example: List All Start Request Forms

This example lists all Start Request Forms:

```bash
wrappers/itpam_call --user=pamadmin --password=pamadmin --endpointaddress=http://lodlc1003.ca.com:8080/itpam/soap --operation=get_start_request_forms
```

The response is as follows:

Form: name=<Update Job Files SRF>, path=/ITPAM Tutorials/WAAE Connector/S05 Update WAAE Job/L03 Update Job Files/

Form: name=<Update WAAE Job SRF>, path=/ITPAM Tutorials/WAAE Connector/S05 Update WAAE Job/L01 Update WAAE Job/

Form: name=<Delete Job SRF>, path=/ITPAM Tutorials/WAAE Connector/S02 Delete WAAE Job/L01 Delete WAAE Job/

...  

Form: name=<Demo_WAAE_1_SRF>, path=/CAWAAE/

Example: Display a Start Request Form

This example displays a Start Request Form named Demo_WAAE_1_SRF in the path /CAWAAE/:

```bash
```

In the following response, the variables and constraints of the Start Request Form are displayed:

Start Request Form: name=Parameters

- element: name=<strName> type=itpamString default=<<>
  - constraint: name=<minlength> value=<> value=0
  - constraint: name=<maxlength> value=2147483647
  - constraint: name=<isReadOnly> value=false
Example: Retrieve the Status of the Process Automation Server

This example retrieves the status of the Process Automation server:

```
wrappers/itpam_call --user=pamadmin --password=pamadmin
--endpointaddress=http://lodlc1003.ca.com:8080/itpam/soap
--operation=check_server_status

The following response indicates that the Process Automation server is in working condition:

Server status: Server status ok.
```
Chapter 4: Running the CLI Using Your Scheduling Manager

You can run the itpam_call and ws_cli scripts using any CA scheduling manager product including CA Workload Automation DE, CA Workload Automation AE, CA Workload Automation ESP Edition, and CA Workload Automation CA 7 Edition. You can then track the job to completion using the standard product features for monitoring.

Note: For more information about using one of these CA scheduling manager products, see the appropriate CA documentation.

This section contains the following topics:

Integration with CA Workload Automation DE (see page 43)
Integration with CA Workload Automation AE (see page 50)
Integration with CA Workload Automation ESP Edition (see page 56)
Integration with CA Workload Automation CA 7 Edition (see page 60)

Integration with CA Workload Automation DE

To run the itpam_call and ws_cli scripts using CA Workload Automation DE, you define a UNIX or Windows job in CA WA Desktop Client that runs the script. The job is defined as part of an Application and runs on a UNIX or Windows agent.

Note: For more information about defining UNIX or Windows jobs using CA Workload Automation DE, see the CA Workload Automation DE Define Perspective Help.
Example: Invoke a Web Service to Get Movie Information Using a UNIX Job

This example invokes a web service to get movie information using a UNIX job. The job requests information on movies playing in theaters within a 20-mile radius of the 12590 zip code. The following CLI arguments are passed to the shell script:

- Web service name (MovieInformation)
- WSDL port name (MovieInformationSoap)
- Operation to invoke (GetTheatersAndMovies)
- Target endpoint address URL (http://www.ignyte.com/webservices/ignyte.whatsshowing.webservice/moviefunctions.asmx)
- Parameters to pass to the web service (zipCode=12590; radius=20)

To invoke a web service to get movie information using a UNIX job

1. Create a UNIX job in the Application.
2. Enter the following information in the Basic page of the job definition:
   - Name—showing
   - Agent—UNIXAGENT
   - Script/command name—wrappers/ws_cli
   - Arguments to pass—
     --svcname=MovieInformation
     --portname=MovieInformationSoap
     --operation=GetTheatersAndMovies
     --endpoint=http://www.ignyte.com/webservices/ignyte.whatsshowing.webservice/moviefunctions.asmx
     --param[\GetTheatersAndMovies]="
     --param[\GetTheatersAndMovies/zipCode]="12590"
     --param[\GetTheatersAndMovies/radius]="20"
3. Select the Run a script option button.
4. Click OK.
Example: Invoke a Web Service to Get Weather Information Using a Windows Job

This example invokes a web service to get weather information for a weather station using a Windows job. The job requests weather information for Raleigh-Durham International Airport, United States. The following CLI arguments are passed to the batch script:

- WSDL URL (http://www.webservicex.net/globalweather.asmx?wsdl)
- Web service name (GlobalWeather)
- WSDL port name (GlobalWeatherSoap)
- Operation to invoke (GetWeather)
- Target endpoint address URL (http://www.webservicex.net/globalweather.asmx)
- Parameters to pass to the web service (CityName=Raleigh / Durham, Raleigh-Durham International Airport; CountryName=United States)

To invoke a web service to get weather information using a Windows job

1. Create a Windows job in the Application.
2. Enter the following information in the Basic page of the job definition:
   - Name—weather
   - Agent—WINAGENT
   - Command to run—wrappers\ws_cli.bat
   - Arguments to pass
     - --wsdl=http://www.webservicex.net/globalweather.asmx?wsdl
     - --svcname=GlobalWeather
     - --portname=GlobalWeatherSoap
     - --operation=GetWeather
     - --endpoint=http://www.webservicex.net/globalweather.asmx
     - --param[/GetWeather][]="" --param[/GetWeather/CityName][]="Raleigh / Durham, Raleigh-Durham International Airport"
     - --param[/GetWeather/CountryName][]="United States"
3. Click OK.

Example: Invoke a CA Workload Automation DE Web Service

You can use the Web Services CLI to invoke any of the web services functions described in the CA Workload Automation DE Programming Guide. You can invoke a CA Workload Automation DE web service over HTTP or HTTPS.

Note: To invoke CA Workload Automation DE web services using the Web Services CLI, you require CA Workload Automation DE r11.3 SP1 or higher.
Example: Invoke a CA Workload Automation DE Web Service over HTTP

This example invokes a CA Workload Automation DE web service to execute a CLI command using a Windows job. The following CLI arguments are passed to the batch script:

- WSDL URL (http://workstation:8080/axis2/services/EspDSeriesService?wsdl)
- Web service name (EspDSeriesService)
- WSDL port name (EspDSeriesServiceHttpSoap11Endpoint)
- Operation to invoke (executeCliCommand)
- Target endpoint address URL (http://workstation:8080/axis2/services/EspDSeriesService)
- Parameters to pass to the web service (command=about)
- Target authentication settings (host=workstation; port=8080; user=schedmaster; authentication order=BASIC; password=0B00E83F785A2461DBF632)

This example assumes that CA WA Web Services is installed on a host named workstation and is listening on the 8080 port. Use the password utility on the agent to generate the encrypted password.

To invoke a CA Workload Automation DE web service over HTTP

1. Create a Windows job in the Application.
2. Enter the following information in the Basic page of the job definition:
   - Name—executeCliCommand
   - Agent—WINAGENT
   - Command to run—wrappers\ws_cli.bat
   - Arguments to pass:
     ```
     --wsdl=http://workstation:8080/axis2/services/EspDSeriesService?wsdl
     --svcname=EspDSeriesService
     --portname=EspDSeriesServiceHttpSoap11Endpoint
     --operation=executeCliCommand
     --endpoint=http://workstation:8080/axis2/services/EspDSeriesService
     --param["/executeCliCommand"]=""
     --param["/executeCliCommand/command"]="about"
     --target_auth[host]=workstation --target_auth[port]=8080
     --target_auth[user]=schedmaster --target_auth[order]=BASIC
     --target_auth[password]=0B00E83F785A2461DBF632
     ```
3. Click OK.
Example: Invoke a CA Workload Automation DE Web Service over HTTPS

This example invokes a CA Workload Automation DE web service to create a global variable in the DEFAULT context using a UNIX job. The following CLI arguments are passed to the batch script:

- Web service name (EspDSeriesService)
- WSDL port name (EspDSeriesServiceHttpSoap11Endpoint)
- Operation to invoke (createVariable)
- Target endpoint address URL (https://workstation:8443/axis2/services/EspDSeriesService)
- Parameters to pass to the web service (name=level; value=2; context=DEFAULT)
- Target authentication settings (host=workstation; port=8443, user=schedmaster; authentication order=BASIC; password=0B00E83F785A2461DBF632)
- Keystore parameters (path=/opt/MIG/ESPSystemAgent/cacerts; password=055A55EB863D2A5D; alias=tomcat)
- Truststore parameters (path=/opt/MIG/ESPSystemAgent/cacerts; password=055A55EB863D2A5D)

This example assumes that CA WA Web Services is installed on a host named workstation and is listening on the 8443 port. Since HTTPS communication is used, the keystore and trust store parameters are required. Use the password utility on the agent to generate the encrypted passwords.

To invoke a CA Workload Automation DE web service over HTTPS

1. Create a UNIX job in the Application.
2. Enter the following information in the Basic page of the job definition:
   - Name—createVariable
   - Agent—UNIXAGENT
   - Command to run—wrappers/ws_cli
Arguments to pass:
--wsdl=https://workstation:8443/axis2/services/EspDSeriesService?wsdl
--svcname=EspDSeriesService
--portname=EspDSeriesServiceHttpSoap11Endpoint --operation=createVariable
--endpoint=https://workstation:8443/axis2/services/EspDSeriesService
--param["/createVariable"]="" --param["/createVariable/name"]="level"
--param["/createVariable/value"]=2
--param["/createVariable/context"]="DEFAULT"
--target_auth[host]=workstation --target_auth[port]=8443
--target_auth[user]=schedmaster
--target_auth[password]=0B00E83F785A2461DBF632
--keystore[password]="055A55EB863D2A5D" --keystore[alias]="tomcat"
--target_auth[order]=BASIC
--truststore[password]="055A55EB863D2A5D"

3. Click OK.
Example: Execute a Process Automation Process Using a UNIX Job

This example executes a Process Automation process using a UNIX job. The following CLI arguments are passed to the shell script:

- Process Automation user name and password (pamadmin)
- Target endpoint address URL on the CA Process Automation server (http://panna01-i24591:8080/itpam/soap)
- Name of the operation (exec_process)
- Fully qualified name of the process (/CAWAE/command)
- Option to track the process to completion

To execute a Process Automation process using a UNIX job
1. Create a UNIX job in the Application.
2. Enter the following information in the Basic page of the job definition:
   - Name—exec_process
   - Agent—UNIXAGENT
   - Script/command name—wrappers/itpam_call
   - Arguments to pass—
     --user=pamadmin
     --password=pamadmin
     --endpointaddress=http://panna01-i24591:8080/itpam/soap
     --operation=exec_process
     --mainarg[name]=/CAWAE/command
     --operation_switch[track]=true
3. Select the Run a script option button.
4. Click OK.
Example: Submit a Start Request Form Using a Windows Job

This example submits a Start Request Form using a Windows job. The following CLI arguments are passed to the batch script:

- Process Automation user name and password (pamadmin)
- Target endpoint address URL on the CA Process Automation server (http://panna01-i24591:8080/itpam/soap)
- Name of the operation (exec_start_request)
- Name of the Start Request Form (SRF_command)
- Path of the Start Request Form (/CAWAAE/)
- Option to track the Start Request Form to completion

To submit a Start Request Form using a Windows job

1. Create a Windows job in the Application.
2. Enter the following information in the Basic page of the job definition:
   - Name—exec_start_request
   - Agent—WINAGENT
   - Command to run—\wrappers\itpam_call.bat
   - Arguments to pass—
     ```
     --user=pamadmin --password=pamadmin
     --endpointaddress=http://panna01-i24591:8080/itpam/soap
     --operation=exec_start_request --mainarg[name]=SRF_command
     --mainarg[path]=/CAWAAE/ --operation_switch[track]=true
     ```
3. Click OK.

Integration with CA Workload Automation AE

To run the itpam_call and ws_cli scripts using CA Workload Automation AE, you define a command job in JIL or CA Workload Control Center that runs the script. The job runs on a UNIX or Windows agent.

Notes:

- You cannot insert a job if the value specified in the command exceeds 512 bytes. To avoid this limitation, embed the entire CLI command in a batch or shell script and specify the full path to the script as the command.
- For more information about defining command jobs using CA Workload Automation AE, see the CA Workload Automation AE User Guide and the CA Workload Automation AE Reference Guide.
- For more information about defining command jobs using CA Workload Control Center, see the CA Workload Control Center Workload Scheduling Guide.
Example: Invoke a Web Service to Get Movie Information Using a Command Job

This example invokes a web service to get movie information using a command job. The job runs on a UNIX agent and requests information on movies playing in theaters within a 20-mile radius of the 12590 zip code. The following CLI arguments are passed to the shell script:

- WSDL URL
  (http://www.ignyte.com/webservices/ignyte.whatsshowing.webservice/moviefunct
   ions.asmx?wsdl)
- Web service name (MovieInformation)
- WSDL port name (MovieInformationSoap)
- Operation to invoke (GetTheatersAndMovies)
- Target endpoint address URL
  (http://www.ignyte.com/webservices/ignyte.whatsshowing.webservice/moviefunct
   ions.asmx)
- Parameters to pass to the web service (zipCode=12590; radius=20)

To define the command job in JIL

Insert a job and specify the following attributes in the definition:

```bash
insert_job: showing
  job_type: CMD
  machine: unixagent
  command: wrappers/ws_cli
  --wsdl="http://www.ignyte.com/webservices/ignyte.whatsshowing.webservice/moviefunct
   ions.asmx?wsdl" --svcname=MovieInformation --portname=MovieInformationSoap
  --operation=GetTheatersAndMovies
  --endpoint="http://www.ignyte.com/webservices/ignyte.whatsshowing.webservice/mov
   efuctions.asmx" --param[/GetTheatersAndMovies]="
  --param[/GetTheatersAndMovies/zipCode]="12590"
  --param[/GetTheatersAndMovies/radius]="20"
```
To define the command job in CA Workload Control Center

1. Create a Command job.

2. Enter the following properties:
   - Name—showing
   - Send to machine—UNIXAGENT
   - Command—wrappers/ws_cli
     --wsdl=http://www.ignyte.com/webservices/ignyte.whatsshowing webservice/moviefunctions.asmx?wsdl --svcname=MovieInformation
     --portname=MovieInformationSoap --operation=GetTheatersAndMovies
     --param[/GetTheatersAndMovies/zipCode]="12590"
     --param[/GetTheatersAndMovies/radius]="20"

3. Commit the job.
Example: Invoke a Web Service to Get Weather Information Using a Command Job

This example invokes a web service to get weather information for a weather station using a command job. The job runs on a Windows agent and requests weather information for Raleigh-Durham International Airport, United States. The following CLI arguments are passed to the batch script:

- WSDL URL (http://www.webservicex.net/globalweather.asmx?wsdl)
- Web service name (GlobalWeather)
- WSDL port name (GlobalWeatherSoap)
- Operation to invoke (GetWeather)
- Target endpoint address URL (http://www.webservicex.net/globalweather.asmx)
- Parameters to pass to the web service (CityName=Raleigh / Durham, Raleigh-Durham International Airport; CountryName=United States)

To define the command job in JIL

Insert a job and specify the following attributes in the definition:

```plaintext
insert_job: weather
job_type: CMD
machine: winagent
command: wrappers\ws_cli.bat
--wsdl="http://www.webservicex.net/globalweather.asmx?wsdl"
--svcname=GlobalWeather --portname=GlobalWeatherSoap --operation=GetWeather
--endpoint="http://www.webservicex.net/globalweather.asmx"
```
To define the command job in CA Workload Control Center

1. Create a Command job.
2. Enter the following properties:
   - Name—weather
   - Send to machine—WINAGENT
   - Command—wrappers\ws_cli.bat
     --wsdl=http://www.webservicex.net/globalweather.asmx?wsdl
     --svcname=GlobalWeather --portname=GlobalWeatherSoap
     --operation=GetWeather
     --endpoint=http://www.webservicex.net/globalweather.asmx
     --param[/GetWeather]="" --param[/GetWeather/CityName]="Raleigh / Durham, Raleigh-Durham International Airport"
     --param[/GetWeather/CountryName]="United States"
3. Commit the job.

Example: Execute a Process Automation Process Using a Command Job

This example executes a Process Automation process using a command job. The job runs on a UNIX agent. The following CLI arguments are passed to the shell script:
   - Process Automation user name and password (pamadmin)
   - Target endpoint address URL on the CA Process Automation server
     (http://panna01-i24591:8080/itpam/soap)
   - Name of the operation (exec_process)
   - Fully qualified name of the process (/CAWAAE/command)
   - Option to track the process to completion

To define the command job in JIL

Insert a job and specify the following attributes in the definition:

```ini
insert_job: exec_process
machine: unixagent
command: wrappers/itpam_call --user=pamadmin --password=pamadmin
       --endpointaddress="http://panna01-i24591:8080/itpam/soap"
       --operation=exec_process --mainarg[name]=/CAWAAE/command
       --operation_switch[track]=true
```
To define the command job in CA Workload Control Center

1. Create a Command job.
2. Enter the following properties:
   ■ Name—exec_process
   ■ Send to machine—UNIXAGENT
   ■ Command—wrappers/itpam_call --user=pamadmin --password=pamadmin
     --endpointaddress=http://panna01-i24591:8080/itpam/soap
     --operation=exec_process --mainarg[name]=/CAWAAE/command
     --operation_switch[track]=true
3. Commit the job.

Example: Submit a Start Request Form Using a Command Job

This example submits a Start Request Form using a command job. The job runs on a Windows agent. The following CLI arguments are passed to the batch script:

■ Process Automation user name and password (pamadmin)
■ Target endpoint address URL on the CA Process Automation server (http://panna01-i24591:8080/itpam/soap)
■ Name of the operation (exec_start_request)
■ Name of the Start Request Form (SRF_command)
■ Path of the Start Request Form (/CAWAAE/)
■ Option to track the Start Request Form to completion

To define the command job in JIL

Insert a job and specify the following attributes in the definition:

```plaintext
insert_job: exec_start_request
job_type: CMD
machine: winagent
command: wrappers\itpam_call.bat --user=pamadmin --password=pamadmin
--endpointaddress=\"http://panna01-i24591:8080/itpam/soap\"
--operation=exec_start_request --mainarg[name]=SRF_command
--mainarg[path]=/CAWAAE/ --operation_switch[track]=true
```
To define the command job in CA Workload Control Center

1. Create a Command job.

2. Enter the following properties:

   ■ Name—exec_start_request
   ■ Send to machine—WINAGENT
   ■ Command—wrappers\itpam_call.bat --user=pamadmin --password=pamadmin
      --endpointaddress=http://panna01-i24591:8080/itpam/soap
      --operation=exec_start_request --mainarg[name]=SRF_command
      --mainarg[path]=/CAWAAE/ --operation_switch[track]=true

3. Commit the job.

Integration with CA Workload Automation ESP Edition

To run the itpam_call and ws_cli scripts using CA Workload Automation ESP Edition, you define a UNIX or Windows job that runs the script. The job is defined as part of an Application and runs on a UNIX or Windows agent.

Note: For more information about defining UNIX or Windows jobs using CA Workload Automation ESP Edition, see the CA Workload Automation Agent for UNIX, Linux, or Windows User Guide.
Example: Invoke a Web Service to Get Movie Information Using a UNIX Job

This example invokes a web service to get movie information using a UNIX job. The job requests information on movies playing in theaters within a 20-mile radius of the 12590 zip code. The following CLI arguments are passed to the shell script:

- WSDL URL
- Web service name (MovieInformation)
- WSDL port name (MovieInformationSoap)
- Operation to invoke (GetTheatersAndMovies)
- Target endpoint address URL
  (http://www.ignyte.com/webservices/ignyte.whatsshowing.webservice/moviefunctions.asmx)
- Parameters to pass to the web service (zipCode=12590; radius=20)

The following job definition invokes the web service to get movie information using a UNIX job:

```
UNIX_JOB SHOWING
  AGENT UNIXAGENT
  CMDNAME wrappers/ws_cli
  ARGS --
  --svcdname=MovieInformation --portname=MovieInformationSoap +
  --operation=GetTheatersAndMovies +
  --endpoint=http://www.ignyte.com/webservices/+ignyte.whatsshowing.webservice/moviefunctions.asmx +
  --param[/GetTheatersAndMovies]="" +
  --param[/GetTheatersAndMovies/zipCode]="12590" +
  --param[/GetTheatersAndMovies/radius]="20"
  RUN DAILY
ENDJOB
```
Example: Invoke a Web Service to Get Weather Information Using a Windows Job

This example invokes a web service to get weather information for a weather station using a Windows job. The job requests weather information for Raleigh-Durham International Airport, United States. The following CLI arguments are passed to the batch script:

- WSDL URL (http://www.webservicex.net/globalweather.asmx?wsdl)
- Web service name (GlobalWeather)
- WSDL port name (GlobalWeatherSoap)
- Operation to invoke (GetWeather)
- Target endpoint address URL (http://www.webservicex.net/globalweather.asmx)
- Parameters to pass to the web service (CityName=Raleigh / Durham, Raleigh-Durham International Airport; CountryName=United States)

The following job definition invokes the web service to get weather information using a Windows job:

```
NT_JOB WEATHER
   AGENT WINAGENT
   CMDNAME wrappers\ws_cli.bat
   ARGS --wsdl=http://www.webservicex.net/globalweather.asmx?wsdl +
          --svcname=GlobalWeather --portname=GlobalWeatherSoap +
          --operation=GetWeather +
          --endpoint=http://www.webservicex.net/globalweather.asmx +
          --param[/GetWeather]="" --param[/GetWeather/CityName]="Raleigh / +
          Durham, Raleigh-Durham International Airport" +
          --param[/GetWeather/CountryName]="United States"
   RUN DAILY
ENDJOB
```
Example: Execute a Process Automation Process Using a UNIX Job

This example executes a Process Automation process using a UNIX job. The following CLI arguments are passed to the shell script:

- Process Automation user name and password (pamadmin)
- Target endpoint address URL on the CA Process Automation server (http://panna01-i24591:8080/itpam/soap)
- Name of the operation (exec_process)
- Fully qualified name of the process (/CAWAAE/command)
- Option to track the process to completion

The following job definition executes the Process Automation process using a UNIX job:

```bash
UNIX_JOB EXEC_PROCESS
  AGENT UNIXAGENT
  CMDNAME wrappers/itpam_call
  ARGS --user=pamadmin --password=pamadmin +
         --endpointaddress=http://panna01-i24591:8080/itpam/soap +
         --operation=exec_process --mainarg[name]=/CAWAAE/command +
         --operation_switch[track]=true
  RUN DAILY
ENDJOB
```
Example: Submit a Start Request Form Using a Windows Job

This example submits a Start Request Form using a Windows job. The following CLI arguments are passed to the batch script:

- Process Automation user name and password (pamadmin)
- Target endpoint address URL on the CA Process Automation server (http://panna01-i24591:8080/itpam/soap)
- Name of the operation (exec_start_request)
- Name of the Start Request Form (SRF_command)
- Path of the Start Request Form (/CAWAAE/)
- Option to track the Start Request Form to completion

The following job definition submits the Start Request Form using a Windows job:

```
NT_JOB EXEC_START_REQUEST
  AGENT WINAGENT
  CMDNAME wrappers\itpam_call.bat
  ARGS --user=pamadmin --password=pamadmin +
  --endpointaddress=http://panna01-i24591:8080/itpam/soap +
  --operation=exec_start_request --mainarg[name]=SRF_command +
  --mainarg[path]=/CAWAAE/ --operation_switch[track]=true
  RUN DAILY
ENDJOB
```

Integration with CA Workload Automation CA 7 Edition

To run the itpam_call and ws_cli scripts using CA Workload Automation CA 7 Edition, you define a UNIX or Windows job that runs the script. The job runs on a UNIX or Windows agent.

**Note:** For more information about defining UNIX or Windows jobs using CA Workload Automation CA 7 Edition, see the CA Integrated Agent Services User Guide.
Example: Invoke a Web Service to Get Movie Information Using a UNIX Job

This example invokes a web service to get movie information using a UNIX job. The job requests information on movies playing in theaters within a 20-mile radius of the 12590 zip code. The following CLI arguments are passed to the shell script:

- WSDL URL
- Web service name (MovieInformation)
- WSDL port name (MovieInformationSoap)
- Operation to invoke (GetTheatersAndMovies)
- Target endpoint address URL
  (http://www.ignyte.com/webservices/ignyte.whatsshowing.webservice/moviefunctions.asmx)
- Parameters to pass to the web service (zipCode=12590; radius=20)

The following job definition invokes a web service to get movie information using a UNIX job:

```
AGENT UNIXAGENT
CMDNAME wrappers/ws_cli
ARGS --
  --svcname=MovieInformation --portname=MovieInformationSoap +
  --operation=GetTheatersAndMovies +
  --endpoint=http://www.ignyte.com/webservices/+ignyte.whatsshowing.webservice/moviefunctions.asmx +
  --param[/GetTheatersAndMovies]="" +
  --param[/GetTheatersAndMovies/zipCode]="12590" +
  --param[/GetTheatersAndMovies/radius]="20"
```
Example: Invoke a Web Service to Get Weather Information Using a Windows Job

This example invokes a web service to get weather information for a weather station using a Windows job. The job requests weather information for Raleigh-Durham International Airport, United States. The following CLI arguments are passed to the batch script:

- WSDL URL (http://www.webservicex.net/globalweather.asmx?wsdl)
- Web service name (GlobalWeather)
- WSDL port name (GlobalWeatherSoap)
- Operation to invoke (GetWeather)
- Target endpoint address URL (http://www.webservicex.net/globalweather.asmx)
- Parameters to pass to the web service (CityName=Raleigh / Durham, Raleigh-Durham International Airport; CountryName=United States)

The following job definition invokes a web service to get weather information using a Windows job:

```
AGENT WINAGENT
CMDNAME wrappers\ws cli.bat
ARGS --wsdl=http://www.webservicex.net/globalweather.asmx?wsdl +
     --svcname=GlobalWeather --portname=GlobalWeatherSoap +
     --operation=GetWeather +
     --endpoint=http://www.webservicex.net/globalweather.asmx +
     --param[/GetWeather]="" --param[/GetWeather/CityName]="Raleigh / Durham, Raleigh-Durham International Airport" +
     --param[/GetWeather/CountryName]="United States"
```
Example: Execute a Process Automation Process Using a UNIX Job

This example executes a Process Automation process using a UNIX job. The following CLI arguments are passed to the shell script:

- Process Automation user name and password (pamadmin)
- Target endpoint address URL on the CA Process Automation server (http://panna01-i24591:8080/itpam/soap)
- Name of the operation (exec_process)
- Fully qualified name of the process (/CAWAAE/command)
- Option to track the process to completion

The following job definition executes the Process Automation process using a UNIX job:

AGENT UNIXAGENT
CMDNAME wrappers/itpam_call
ARGS --user=pamadmin --password=pamadmin + --endpointaddress=http://panna01-i24591:8080/itpam/soap + --operation=exec_process --mainarg[name]=/CAWAAE/command + --operation_switch[track]=true

Example: Submit a Start Request Form Using a Windows Job

This example submits a Start Request Form using a Windows job. The following CLI arguments are passed to the batch script:

- Process Automation user name and password (pamadmin)
- Target endpoint address URL on the CA Process Automation server (http://panna01-i24591:8080/itpam/soap)
- Name of the operation (exec_start_request)
- Name of the Start Request Form (SRF_command)
- Path of the Start Request Form (/CAWAAE/
- Option to track the Start Request Form to completion

The following job definition submits the Start Request Form using a Windows job:

AGENT WINAGENT
CMDNAME wrappers\itpam_call.bat

Chapter 4: Running the CLI Using Your Scheduling Manager  63