Stonebranch Solutions

Indesca
CA 7® Interface Using SOAP

indesca-ca7-soap

Document Name	Indesca CA 7® Interface Using SOAP
Document ID	indesca-ca7-soap
Products	Indesca 4.1.0

Stonebranch, Inc.
950 North Point Parkway, Suite 200
Alpharetta, GA 30005 USA
Tel: (678) 366-7887
Fax: (678) 366-7887

Stonebranch, Inc. does not provide support for CA 7® Workload Automation.
For technical information or instruction on CA 7® operations, contact CA Technologies.

SAP Certified Integration

Contents

Figures

Overview

This document is intended for an enterprise with a requirement to interface applications with CA 7 using the Simple Object Access Protocol (SOAP). This interface will be bidirectional, meaning that CA 7 will initiate application processes and the applications will initiate batch jobs in CA 7.

The Indesca implementation of this uses Universal Command Agent for SOA and Universal Event Monitor for SOA.

This document provides an overview of the implementation and the specific configurations and setups required to implement this solution.

Indesca Overview

Indesca (Independent Scheduling Agents) is a workload agent that integrates with any job scheduling engine, enabling standardized system-wide processes and procedures.

Indesca Components

The following Indesca components have been utilized in this solution:

- Universal Command
- Universal Command Agent for SOA
- Universal Event Monitor
- Universal Event Monitor for SOA

CA 7 Overview

CA Technologies' CA 7 Workload Automation (CA 7) is a mainframe-hosted job scheduling product.

SOAP Protocol Overview

SOAP is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks.

It relies on eXtensible Markup Language (XML) as its message format and usually relies on other Application Layer protocols (most notably Remote Procedure Call (RPC) and HTTP) for message negotiation and transmission.

Implementation Description

Overview

This Indesca SOAP implementation consists of an installation of the Indesca Agent and the Indesca Universal Command Agent for SOA agent on a Linux Server.

The Indesca Agent is also installed on the z/OS mainframe LPAR that hosts CA 7.

Outbound SOAP Requests

Outbound SOAP requests are made from CA 7 via a CA 7-submitted batch job. The CA 7-submitted batch job utilizes Universal Command to initiate the Universal Command Agent for SOA on the Linux server.

The SOAP method used is request / acknowledge. Request / acknowledge means that the batch job completes once it has received an acknowledgement from the application that the delivered SOAP message has been received. At this point, neither Indesca nor CA 7 is aware of the status of any application processes initiated by the delivered SOAP message.

The applications publish status via an Inbound SOAP request.

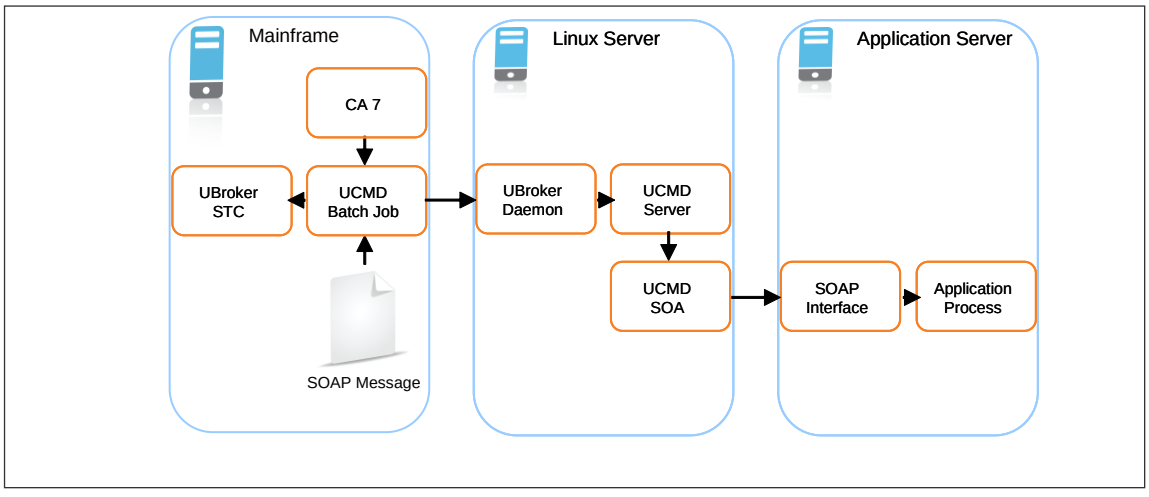

Figure 1 Outbound SOAP Requests

Inbound SOAP Requests

Inbound SOAP requests are handled via the Universal Event Monitor for SOA listener.

Once the listener detects an inbound SOAP message, it writes the message payload to a file. Universal Event Monitor (a component of the Indesca Agent) detects the file and initiates an action.

The SOAP message payload is parsed to extract information that is used to build a z/OS console message. Universal Command delivers the message from the Linux server to the z/OS mainframe. Based on the content of the console message, automation is in place to initiate any required workload in CA 7.

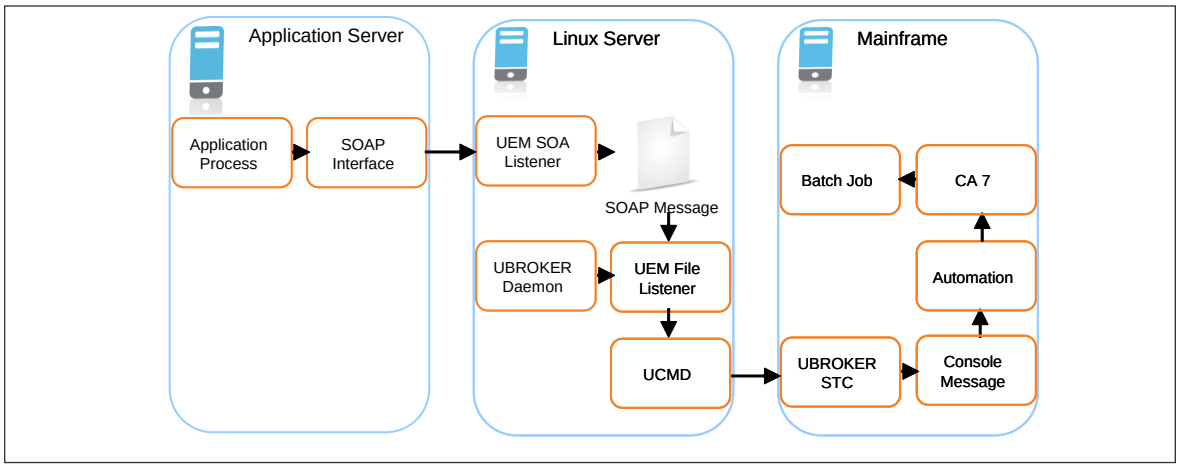

Figure 2 Inbound SOAP Requests

Outbound SOAP Implementation

The outbound SOAP message delivered to the application contains the following three parameters:

1. Run Date
   Today's date in the format YYYY-MM-DD. This will be generated in CA7.
2. Request Identifier
   A request identifier will be provided by the application.
3. Run Type
   At this time, 'START' is the only valid value. In future versions of this project, there is the assumption of other values being sent in this field.

The following JCL will initiate the outbound SOAP request.

```
//TZE025R2 JOB (TEST,CC0KG1500000),'WINDOWS',
JOB08030
//             CLASS=S,
//             MSGCLASS=R
//*
// JCLLIB ORDER=TEST.SYS5.UNV.SUNVSAMP
//*
*********************************************************************
//* * Sample SOA Communication for R1
//* *
*********************************************************************
//* * STEPS - FUNCTION
//* * -----   ---------------------------------------------------------
//* * SYSIN - Target destination for process / LINUX
//* * INPUT - Universal Command Options to execute SOAP
//* * UNVIN - PAYLOAD being passed to server
//*
*********************************************************************
//STEP1    EXEC UCMDPRC
//LOGIN    DD  DISP=SHR,DSN=ZE025.PROD.INDESCA(IDNPSWD)
//SYSIN    DD  DISP=SHR,DSN=XXX.CONTROL.UPARMLIB(HOSTPARM)
//INPUT    DD  DISP=SHR,DSN=XXX.CONTROL.UPARMLIB(SOAPCALL)
//UNVIN    DD  DISP=SHR,DSN=XXX.CC030210.PMS002.STGXML.START
```

Figure 3 Outbound SOAP Requests – Universal Command Batch Job JCL

This JCL executes the Universal Command JCL procedure.

The DD Statements contain the following:

- LOGIN DD
  Encrypted password for the Linux Server running the Universal Command Agent for SOA. The encrypted file is created with the Universal Encrypt utility provided on all platforms with Indesca.

- SYSIN DD
  Universal Command runtime parameters:
  - -HOST
    DNS name or IP address of the Linux Server running the Universal Command Agent for SOA.
  - -ENCRYPTEDFILE
    Specified the DD name that will contain the encrypted password file.
  - -SCRIPT
    Specifies the DD name that will contain the Universal Command Agent for SOA runtime parameters that are passed to the Universal Command Agent for SOA.
  - –SCRYPT_TYPE
    The value SERVICE tells Universal Command that this is a SOA process.

```
-HOST         deveis01
-ENCRYPTEDFILE      LOGIN
-SCRIPT       INPUT
-SCRIPT_TYPE  SERVICE
```

Figure 4 Outbound SOAP Requests – Universal Command Batch Job SYSIN DD Statements

- INPUT DD
  Universal Command Agent for SOA runtime parameters
  - –PROTOCOL
    Indicates which of the supported SOA protocols to use for this request.
  - –MEP
    The value REQUEST tells the Universal Command Agent for SOA that this request is synchronous; that is, two-way and blocked until a reply is sent by the target workload.
  - –SERVICEURL
    Specifies the URL address (internet, network, or file-based) of the target workload.
  - –SERVICEUSERNAME
    Specifies the user name to be passed to the target workload for authentication.
  - –SERVICEPASSWORD
    Specifies the password to be passed to the target workload for authentication.
  - –TIMEOUTSEC
    Specifies the length of time – in seconds – to wait for the request to complete.

```
-protocol SOAP
-mep request
-serviceurl http://asmws2/rbs_ws/services/BatchCtrlSvcWS
-serviceusername dummy
-servicepassword dummy
-timeoutsec 120
```

Figure 5 Outbound SOAP Requests – Universal Command Batch Job SYSIN DD Contents

- UNVIN DD
  Universal Command Agent for SOA payload. Contains the values for Run Date, Request Identifier and Request Type.

```
<est:processBatchCtrlSvcTxn
 xmlns:est="http://xxx.com//services/establish-task-facade/">
        <batchctrlsvcReq>
           <ReqHeader>
              <ReqId>AUT4510021710113870000200</ReqId>
              <CmdType>request</CmdType>
              <CmdMode>alwaysRespond</CmdMode>
              <UserId></UserId>
              <Passwd></Passwd>
           </ReqHeader>
           <BatchCtrlSvc_ReqRecord>
              <Action>START</Action>
              <EODDt>2010-02-17</EODDt>
           </BatchCtrlSvc_ReqRecord>
        </batchctrlsvcReq>
     </est:processBatchCtrlSvcTxn>
```

Figure 6 Outbound SOAP Requests – Universal Command Batch Job SYSIN DD Contents

Inbound SOAP Implementation

The Universal Event Monitor for SOA is configured via the **/etc/universal/UAC.xml** file.

```
<?xml version="1.0" encoding="UTF-8"?>
<sb:UAC xmlns:sb="http://com.stonebranch/UAC/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://com.stonebranch/UAC/ UAC.xsd"/
  <!-- $Id$ -->
  <sb:SOAPConnection>
    <sb:URI>/axis2/services/UACInbound</sb:URI>
    <sb:Listeners>
      <sb:SOAPListener>
        <sb:Operation>process</sb:Operation>
        <sb:Actions>
          <sb:SOAPFileWriter>
<sb:Directory>/export/home/control/indesca/soap_listener/</sb:Directory
>
            <sb:FilenamePattern>process_%Seq%.xml</sb:FilenamePattern>
            <sb:StartSequenceNumber>1</sb:StartSequenceNumber>
            <sb:WriteEnvelope>true</sb:WriteEnvelope>
          </sb:SOAPFileWriter>
        </sb:Actions>
      </sb:SOAPListener>
    </sb:Listeners>
  </sb:SOAPConnection>
</sb:UAC>>
```

Figure 7 Inbound SOAP Requests – Universal Event Monitor for SOA UAC.xml

- Additional SOAP connections can be defined to the UAC.xml if required.
- The Universal Event Monitor for SOA listener writes the payload of the inbound SOAP message to the following directory / file mask:

  /export/home/control/indesca/soap_listener/process_%Seq%.xml
- The variable **%Seq%** is resolved to a sequence number generated by the Universal Event Monitor. The sequence number is incremented by one for each file created and is reset to 1 each time the listener is started.

The following shows an example of the inbound message payload written to the **process_%Seq%.xml** file:

```
<?xml version='1.0' encoding='utf-8'?><soapenv:Envelope xmlns:soapenv=

"http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/

XMLSchema-instance"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soapenv:Body><NS1:process
xmlns:NS1="http://inbound.uac.stonebranch.com"><NS1:identitySourceAppli
cationId>RBS</NS1:identitySourceApplicationId><NS1:identitySourceUserId
/>

<NS1:identitySourcePassword /><NS1:identitySourceToken />

<NS1:activityRequestId>AUT421002110926597029300</NS1:activityRequestId
><NS1:activityStatus>PROCESS CLOSE ACCOUNTING YYYY
MM</NS1:activityStatus>

<NS1:activityState>ACCOUNTING MONTH CLOSING
INPROGRESS</NS1:activityState>

<NS1:activityStateReason>INFO</NS1:activityStateReason><NS1:activityAct
ion>ODPT0001</NS1:activityAction><NS1:activityStartDate>2010-02-
24</NS1:activity

StartDate><NS1:activityStartTime>08:35:42.397382</NS1:activityStartTime
></NS1:process></soapenv:Body></soapenv:Envelope>
```

Figure 8 Inbound SOAP Requests – Universal Event Monitor for SOAP process_%Seq%.xml

The three bolded fields in **process_%Seq%.xml** are used to create the z/OS console message:

- identitySourceApplicationId
- activityRequestId
- activityAction

The Universal Event Monitor detects the file created by the Universal Event Monitor for SOA using the following event definition:

```
BEGIN_EVENT
 EVENT_ID         "XXX SOA EVENT"
 EVENT_TYPE       FILE
 COMP_NAME        UEMS
 STATE            ENABLE
 TRACKING_INT     10
 TRIGGERED_ID     "XXX SOA HANDLER"
 FILESPEC         "/export/home/ control/indesca/soap_listener/*.*"
 MIN_FILE_SIZE    0
 RENAME_FILE      YES
 RENAME_FILESPEC "/export/home/
control/indesca/soap_listener/$(origname).$(origext)"
END_EVENT
```

Figure 9 Inbound SOAP Requests – Universal Event Monitor Event Definition

- The event definition is loaded to the Universal Event Monitor using the following command issued on the Linux server running the Universal Command Agent for SOA.

  /opt/universal/bin/uemload -add -deffile *event_definition.txt*

  Alternatively, changes to the event definition can be effected using the following command:

  /opt/universal/bin/uemload -update -deffile *event_definition.txt*

- The event definition 'moves' each **process_%Seq$.xml** file to a staging directory and invokes the XXX SOA HANDLER.

The following Universal Event Monitor Handler definition processes each **process_%Seq%.xml** file.

```
BEGIN_HANDLER
 HANDLER_ID      "XXX SOA HANDLER"
 ACTION_TYPE     CMD
 MAXRC           0
 USERID          "control"
 PWD             "UACL"
 BEGIN_SCRIPT
  STMT "#!/usr/bin/ksh"
  STMT "exec > /export/home/control/indesca/xxx.log 2>&1"
  STMT "set -xv"
  STMT "/opt/universal/bin/ucmd -script
/export/home/control/indesca/xxx.rexx \"
  STMT "< $UEMRENAMEDFILE \"
  STMT "-HOST mvstcp5 -USERID CTLMNT -PWD UACL \"
  STMT ">> /export/home/control/indesca/xxx.log \"
  STMT "2>&1"
  STMT "if [ $? -gt 0 ]"
  STMT " then"
  STMT "  mv $UEMRENAMEDFILE $UEMORIGFILE"
  STMT " else"
  STMT "  rm $UEMRENAMEDFILE"
  STMT "fi"
  STMT "exit $rc"
 END_SCRIPT
END_HANDLER
```

Figure 10 Inbound SOAP Requests – Universal Event Monitor Handler Definition

- The Event Handler executes under the authority of the USERID control. To allow this userid to authenticate without a password the following UACL definitions were made to `/etc/universal/uacl.conf`.

  ```
  uem_handler control,allow,noauth
  ```

  Changes the Indesca configuration files require the Universal Broker to be refreshed. This can be done with the Universal Control utility.

- The Event Handler invokes Universal Command to:
  - Connect to the z/OS mainframe.
  - Executes a REXX script to parse the required information from `process_%Seq%.xml`.
  - Execute Universal Write-to-Operator to write the required console message.

- The Event Handler appends logging information to `/export/home/control/indesca/xxx.log`
- If the Event Handler does not complete successfully, `process_%Seq%.xml` is moved back to its original location so that processing can be retried. Otherwise, this file is deleted.

The REXX script executed by the Event Handler is stored on the Linux server running the Universal Command Agent for SOA.

```
/* REXX */
TRACE R
  XXX.XML = LINEIN()

  parse value XXX.XML with "<NS1:activityAction>" XXX.ACTN
"</NS1:activityAction>"

  parse value XXX.XML with "<NS1:identitySourceApplicationId>"
XXX.APID "</NS1:identitySourceApplicationId>"

  parse value XXX.XML with "<NS1:activityRequestId>" XXX.RQID
"</NS1:activityRequestId>"

  XXX.UWTO = "EIEOSRAT "XXX.ACTN XXX.APID XXX.RQID

  '/usr/lpp/universal/bin/uwto -msg "'XXX.UWTO'"'
  XXX.RC = RC

  EXIT XXX.RC
```

Figure 11 Outbound SOAP Requests – xxx.rexx

- The REXX script is executed under the z/OS USS environment under the authority of the USERID CTLMNT. To allow this userid to authenticate without a password the following UACL definitions were made to **TEST.SYS5.UNV.UNVCONF(ACLCFG00)**.

  ucmd_access ALL,*,CTLMNT,allow,noauth

- Changes the Indesca configuration files require the Universal Broker to be refreshed. This can be done with the Universal Control utility.

- The REXX script executes the Universal Write-to-Operator utility (uwto) in order to write the required message to the z/OS console.

The xxx.log file is appended to each time a **process_%Seq%.xml** is processed. This file is useful as an audit trail and for problem diagnosis.

In order to ensure that this file does not grow to an unreasonable size, an additional Universal Event Monitor Event and Handler have been implemented to purge this file when it reaches 10mb in size.

```
BEGIN_EVENT
 EVENT_ID         "XXX LOG FILE CLEANUP"
 EVENT_TYPE       FILE
 COMP_NAME        UEMS
 STATE            ENABLE
 TRACKING_INT     10
 TRIGGERED_ID     "XXX LOG FILE CLEANUP"
 FILESPEC         "/export/home/control/indesca/xxx.log"
 MIN_FILE_SIZE    10M
END_EVENT

BEGIN_HANDLER
 HANDLER_ID       "XXX LOG FILE CLEANUP"
 ACTION_TYPE      CMD
 MAXRC            0
 USERID           "control"
 PWD              "UACL"
 CMD              "rm /export/home/control/indesca/xxx.log"
END_HANDLER
```

Figure 12 Outbound SOAP Requests – Event and Handler to Purge xxx.log

950 North Point Parkway, Suite 200
Alpharetta, Georgia 30005
U.S.A.