IBM Cúram Social Program Management

Event Adaptor Guide

Version 6.0.5

IBM Cúram Social Program Management

Event Adaptor Guide

Version 6.0.5

Revised: May 2013

This edition applies to IBM Cúram Social Program Management v6.0 5 and to all subsequent releases unless otherwise indicated in new editions.

Contents

Figures v

Tables vii

Chapter 1. Introduction 1
1.1 Purpose 1
1.2 Target Audience 1
1.3 Pre-Requisites 1
1.4 What is the Intelligent Operations Center? . . . 1
1.5 What is the Event Adaptor? 1
1.6 Using The Event Adaptor 2

Chapter 2. Developing code to use the Event Adaptor 3
2.1 Overview 3
2.2 Configuring External Event Types 3
2.3 Making calls to service-layer APIs 5

Chapter 3. WebSphere Message Broker for Cúram 9
3.1 Overview 9
3.2 What is WebSphere Message Broker? 9
3.3 WebSphere Message Broker Set up for Cúram . . 9

Chapter 4. Administering the Event Adaptor 13
4.1 Overview 13
4.2 Enabling Event Adaptor 13
4.3 Enabling and Disabling External Event Types . . 13
4.4 Configuring a Target System 14

Chapter 5. Appendix 15
5.1 External Links 15

Notices 17
Trademarks 19

Figures

1. Sample CAP Event 2
2. Adding Event Type Code Table Items with CTX data 3
3. Configuring External Event Types using DMX Data 4
4. Using the SimplePublisher 6
5. Using the CAPPublisher 7
6. Using the XMLPublisher 8
7. Sample CAP Event with SOAP envelope 10
8. Sample XSL Transformation 11

Tables

1. Default CAP Event Values 5

Chapter 1. Introduction

1.1 Purpose

This document introduces the functionality offered by the Cúram Event Adaptor. It describes how to develop code to use the Event Adaptor and provides information on how it can be configured by administrators.

1.2 Target Audience

The document is intended to be used by developers and administrators that want to use the Cúram Event Adaptor to send events to the Intelligent Operations Center.

1.3 Pre-Requisites

This document assumes that the reader is familiar with developing and administering Cúram applications, with developing and administering the Intelligent Operations Center, and with developing and administering IBM WebSphere Message Broker.

1.4 What is the Intelligent Operations Center?

The Intelligent Operations Center for Smarter Cities provides an executive dashboard to help city leaders gain insight into all aspects of the city. The executive dashboard spans agencies and enables drill-down capability into each underlying agency such as emergency management, public safety, social services, transportation, or water. Intelligent Operations Center allows city leaders to:

- Leverage information across all city agencies and departments to make smarter decisions
- Anticipate problems to minimize the impact of disruptions to city services and operations
- Coordinate cross-agency resources to respond to issues rapidly and effectively

Intelligent Operations Center is designed to:

- Monitor agency and citywide operations
- Involve citizens and businesses in incident reporting and resolution
- Gather and analyze citizen feedback using social media
- Manage a broad range of government and commercial operations
- Deploy rapidly with minimal IT resources

More information can be found on the Intelligent Operations Center product web site - see Appendix for details.

1.5 What is the Event Adaptor?

Event Adaptor

Cúram contains the Event Adaptor. The Event Adaptor provides a mechanism for Cúram to publish events to the Intelligent Operations Center over Web Services via the Intelligent Operations Center's WebSphere Message Broker instance.

What does the Event Adaptor do?

The Event Adaptor presents an API which allows for data to be supplied to the Intelligent Operations Center. When the API is invoked, a CAP v1.2 (Common Alerting Protocol) event will be created and sent

via WebServices to the Intelligent Operations Center's instance of WebSphere Message Broker. The event can then be routed by WebSphere Message Broker into the Intelligent Operations Center event flow.

Common Alerting Protocol v1.2

The Common Alerting Protocol is an OASIS Standard format for alerts and notifications. More information on CAP v1.2 can be found on the OASIS website - see Appendix for details. An example CAP Event is provided below:

```
<cap:alert xmlns:cap="urn:oasis:names:tc:emergency:cap:1.2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schem
  <cap:identifier>d2a42209-33a5-4fcc-8658-0fc2c08c95da</cap:identifier>
  <cap:sender>IBMCuramSample</cap:sender>
  <cap:sent>2012-07-12T15:11:14+01:00</cap:sent>
  <cap:status>Actual</cap:status>
  <cap:msgType>Alert</cap:msgType>
  <cap:scope>Public</cap:scope>
  <cap:code>KPI</cap:code>
  <cap:info>
    <cap:category>Other</cap:category>
    <cap:event>ea.ipa.pd</cap:event>
    <cap:urgency>Unknown</cap:urgency>
    <cap:severity>Moderate</cap:severity>
    <cap:certainty>Observed</cap:certainty>
    <cap:headline>Sample CAP Event</cap:headline>
    <cap:description>Sample CAP Event Description</cap:description>
    <cap:onset>2012-07-12T15:11:14+01:00</cap:onset>
    <cap:senderName>IBMCuramSample</cap:senderName>
    <cap:parameter>
      <cap:valueName>DisposalDateTime</cap:valueName>
      <cap:value>2012-07-12 15:11:14</cap:value>
    </cap:parameter>
    <cap:parameter>
      <cap:valueName>DisposalDuration</cap:valueName>
      <cap:value>0</cap:value>
    </cap:parameter>
    <cap:parameter>
      <cap:valueName>ProgramType</cap:valueName>
      <cap:value>reference</cap:value>
    </cap:parameter>
    <cap:parameter>
      <cap:valueName>DisposalResult</cap:valueName>
      <cap:value>Deny</cap:value>
    </cap:parameter>
  </cap:info>
</cap:alert>
```

Figure 1. Sample CAP Event

1.6 Using The Event Adaptor

Using the Event Adaptor to publish events to the Intelligent Operations Center involves the following tasks:

1. Develop code to publish the CAP events to the Intelligent Operations Center via the Event Adaptor API. This is described in the "Developing Code to use the Event Adaptor" section.
2. Configure the Intelligent Operations Center WebSphere Message Broker instance to expose a Web Services end-point to receive the messages from the Cúram Intelligent Operations Center Event Adaptor, and route them into the Intelligent Operations Center CAP event flow. This is described in the "Configuring the Intelligent Operations Center WebSphere Message Broker" section.
3. Enable the Event Adaptor component and event types, and configure the Event Adaptor to point to the Intelligent Operations Center WebSphere Message Broker Web Services end-point. This is described in the "Administering the Event Adaptor section.

Chapter 2. Developing code to use the Event Adaptor

2.1 Overview

This chapter outlines how to develop code to use the Event Adaptor to publish CAP messages to the Intelligent Operations Center via it's WebSphere Message Broker Web Services endpoint. This involves the following:

1. Create Codetable Items and Event Types for the events to be sent to the Intelligent Operations Center.
2. Develop code to invoke on the Event Adaptor.

2.2 Configuring External Event Types

External Event Types will need to be configured so that it is possible to manage which events cause CAP messages to be sent to the Message Broker.

Also, Code table items need to be added for each External Event Type so each Event Type will have a code table value associated with it.

For more details on Code Tables, refer to the Application Configuration section of the System Configuration guide.

Adding Event Type Code Table Items with CTX

CTX data files can be used to add External Event Types, as illustrated in the following example.

```
<codetables package="curam.codetable">
  <codetable
    java_identifier="EXTERNALEVENTTYPE"
    name="ExternalEventType"
  >
    <displaynames>
      <locale language="en">External Event Type</locale>
    </displaynames>
    <code
      default="false"
      java_identifier="EVENTADAPTOR_SAMPLE"
      status="ENABLED"
      value="ET2000"
    >
      <locale
        language="en"
        sort_order="0"
      >
        <description>Sample External Event</description>
        <annotation/>
      </locale>
    </code>
  </codetable>
</codetables>
```

Figure 2. Adding Event Type Code Table Items with CTX data

Adding Event Type Code Table Items

Alternatively, another mechanism for providing the external event information is via the admin application UI.

1. Login to the application as a user with system administrative privileges and navigate to the "Code Tables" section.
2. Search for "ExternalEventType". This should return the "External Event Type" code table.
3. Click "New Item..." in the list row menu.
4. On the "New Code Table Item" modal, enter the Item name and the code table value and click "Save". Repeat as necessary.
5. Then click "Publish..." on the "Code Tables" tab menu to make the changes available.

The External Event Type code table item has been added and is now available to be configured.

Configuring External Event Types with DMX

Alternatively, DMX data can be used to configure External Event Types, as illustrated in the following example.

```
<table name="EXTERNALEVENTTYPE">
  <column name="externalEventType" type="text" />
  <column name="externalEventEnabled" type="bool" />
  <column name="externalEventTypeID" type="id" />
  <column name="versionNo" type="number" />
  <row>
    <attribute name="externalEventType">
      <value>ET2000</value>
    </attribute>
    <attribute name="externalEventEnabled">
      <value>1</value>
    </attribute>
    <attribute name="externalEventTypeID">
      <value>8897986963777257472</value>
    </attribute>
    <attribute name="versionNo">
      <value>2</value>
    </attribute>
  </row>
</table>
```

Figure 3. Configuring External Event Types using DMX Data

Configuring External Event Types through the Application

Another mechanism for providing the external event information is via the admin application UI. For details on enabling / disabling, please see the administration section.

To access the External Event Type admin page, login as a user with administrative privileges and navigate to the "Administration Workspace". Select "Event Adaptor" from the shortcuts panel. This will open the Event Adaptor tab.

Adding External Event Types

An External Event Type will need to be added to the list so that it can be managed.

1. Select "Add..." from the page level menu to open the "Add External Event Type" modal.
2. Check the boxes on the events that you would like to add on this modal page, and then click "Yes".

The selected External Event Types will be added to the list.

Deleting External Event Types

An External Event Type may need to be deleted.

1. Select "Delete" from the list row menu of the External Event to be deleted.
2. Clicking on the "Yes" button on the "Delete External Event Type" modal will delete the External Event Type.

2.3 Making calls to service-layer APIs

There are three API methods available to invoke on the Cúram Event Adaptor to create a CAP message and publish it the Intelligent Operations Center's WebSphere Message Broker instance.

SimplePublisher

```
curam.eventadaptor.publishers.intf.SimplePublisherIntf.publish(SimpleHeader,
EventAdaptorNVPairList, EXTERNALEVENTTYPEEntry)
```

The SimplePublisher interface allows for publication of CAP messages with many of the values populated using defaults. This method assigns the values from the SimpleHeader and default values to a CAPHeader. This CAPHeader is then passed as a parameter to the `curam.eventadaptor.publishers.intf.CAPPublisherIntf.publish` method.

The SimpleHeader allows for some values in the CAP message to be set. The values that can be set are:

- Description
- Event
- Headline
- Sender
- Onset

The remainder are default values, set from environment variables (which are defined in the code tables).

Table 1. Default CAP Event Values

Value Name	**Default Value**
Identifier	Random value
Category	"Event"
Code	"Other"
Certainty	"Observed"
Message type	"Alert"
Scope	"Public"
Severity	"Moderate"
Status	"Actual"
Urgency	"Unknown"

The code snippet below illustrates how to use the `SimplePublisherIntf`.

```
@Inject
private TargetSystemDAO targetSystemDAO;

@Inject
SimplePublisher simplePublisher;

public ClassConstructor(string arg0) {
  super(arg0);
  GuiceWrapper.getInjector().injectMembers(this);
}

public simplePublisherSample() {
  // How to set up the variables that are passed to the publish method

  SimpleHeader simpleHeader = new SimpleHeader();

  simpleHeader.description = "Sample description";
  simpleHeader.event = "Sample event title";
  simpleHeader.headline = "Sample headline";
  simpleHeader.sender = "Sample sender";
  simpleHeader.onset = header. DateTime.getCurrentDateTime();

  EventAdaptorNVPair nvPair1 = new EventAdaptorNVPair();
  nvPair1.name = "parameter1 name";
  nvPair1.value = "parameter1 value";

  EventAdaptorNVPair nvPair2 = new EventAdaptorNVPair();
  nvPair2.name = "parameter2 name";
  nvPair2.value = "parameter2 value";

  nvPairList.dtls.add(nvPair1);
  nvPairList.dtls.add(nvPair2);

  // Call the publish method
  simplePublisher.publish(simpleHeader, nvPairList, EXTERNALEVENTTYPEEntry.EVENTADAPTOR_SAMPLE);
}
```

Figure 4. Using the SimplePublisher

CAPPublisher

`curam.eventadaptor.publishers.intf.CAPPublisherIntf.publish(CAPHeader, EventAdaptorNVPairList, EXTERNALEVENTTYPEEntry)`

The CAPPublisher interface allows for publication of CAP messages with all of the values populated by the values passed in in the CAPHeader parameter.

This method validates the CAPHeader. The content of the header is then used to build a CAP XML document using `capXMLBuilder.convertToDocument()`. The output from this conversion is then passed as a parameter to the `curam.eventadaptor.publishers.intf.XMLPublisherIntf.publish` method.

The code snippet below illustrates how to use the `CAPPublisherIntf`.

```
@Inject
private TargetSystemDAO targetSystemDAO;

@Inject
CAPPublisher capPublisher;

public ClassConstructor(string arg0) {
  super(arg0);
  GuiceWrapper.getInjector().injectMembers(this);
}

public capPublisherSample() {
  // How to set up the variables that are passed to the publish method

  CAPHeader capHeader = new CAPHeader();

  capHeader.description = "Sample description";
  capHeader.event = "Sample event name";
  capHeader.headline = "Sample headline";
  capHeader.identifier = "sampleIdentifier";
  capHeader.sender = "Sample sender";
  capHeader.category = EAALERTINFOCATEGORY.FIRE;
  capHeader.certainty = EAALERTINFOCERTAINTY.LIKELY;
  capHeader.messageType = EAALERTMESSAGETYPE.UPDATE;
  capHeader.onset = DateTime.getCurrentDateTime();
  capHeader.senderName = "the sendername";
  capHeader.scope = EAALERTSCOPE.PRIVATE;
  capHeader.severity = EAALERTINFOSEVERITY.EXTREME;
  capHeader.status = EAALERTSTATUS.TEST;
  capHeader.urgency = EAALERTINFOURGENCY.EXPECTED;
  capHeader.code = EAALERTCODE.EVENT;
  capHeader.webIdentifier = "http://www.curamsoftware.com";

  EventAdaptorNVPairList nvPairList = new EventAdaptorNVPairList();

  EventAdaptorNVPair nvPair1 = new EventAdaptorNVPair();
  nvPair1.name = "parameter1 name";
  nvPair1.value = "parameter1 value";

  EventAdaptorNVPair nvPair2 = new EventAdaptorNVPair();
  nvPair2.name = "parameter2 name";
  nvPair2.value = "parameter2 value";

  nvPairList.dtls.add(nvPair1);
  nvPairList.dtls.add(nvPair2);

  // Call the publish method
  capPublisher.publish(capHeader, nvPairList, EXTERNALEVENTTYPEEntry.EVENTADAPTOR_SAMPLE);
}
```

Figure 5. Using the CAPPublisher

XMLPublisher

`curam.eventadaptor.publishers.intf.XMLPublisherImpl.publish(Document, EXTERNALEVENTTYPEEntry)`

The XMLPublisher interface allows for publication of SOAP messages with XML Document passed in in the Document parameter. This gives the most control of any of the publish APIs, as the XML is custom. This is the method that will be used if the Event Adaptor is to interact with an external service which does not work with CAP messages, or if a custom crafted CAP message is preferred.

If the External Event Type is enabled, it will validate the XML and then publish the document to the web service endpoint. If the Event Type is not enabled, then it will not publish the document.

The code snippet below illustrates how to use the XMLPublisherIntf.

```
@Inject
private TargetSystemDAO targetSystemDAO;

@Inject
XMLPublisher xmlPublisher;

public ClassConstructor(string arg0) {
  super(arg0);
  GuiceWrapper.getInjector().injectMembers(this);
}

public xmlPublisherSample(Document document) {
  // Call the publish method
  XmlPublisher.publish(document, EXTERNALEVENTTYPEEntry.EVENTADAPTOR_SAMPLE);
}
```

Figure 6. Using the XMLPublisher

Chapter 3. WebSphere Message Broker for Cúram

3.1 Overview

This chapter describes the steps involved in configuring WebSphere Message Broker to enable it to receive, process, and send on the CAP events created by the Cúram Event Adaptor.

3.2 What is WebSphere Message Broker?

WebSphere Message Broker is a powerful information broker that allows both business data and information, in the form of messages, to flow between disparate applications and across multiple hardware and software platforms. Business rules can be applied to the data that is flowing through the message broker in order to route, store, retrieve, and transform the information.

3.3 WebSphere Message Broker Set up for Cúram

Process Overview

The Cúram Event Adaptor publishes events to a SOAP endpoint. The WebSphere Message Broker installation which forms part of the Intelligent Operations Center can be used to expose a SOAP endpoint reference in order to receive the messages published by the Cúram Event Adaptor.

Messages published by the Cúram Event Adaptor to the SOAP endpoint are in a slightly different format to that required by the Intelligent Operations Center. For example, the event is wrapped in a SOAP envelope. A sample CAP event as published by the Cúram Event Adapter is supplied below. WebSphere Message Broker can also be used to extract the CAP message from the SOAP envelope, and remove the other extra elements so that it is in the precise CAP message format required by Intelligent Operations Center. Finally, WebSphere Message Broker can be used to forward the transformed CAP message to the Intelligent Operations Center by routing the messages to the appropriate Intelligent Operations Center input queue.

Sample CAP Event

The following is a sample CAP event with a SOAP envelope as sent by the Cúram Event Adaptor:

```
<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
  <soapenv:Body>
    <ns1:publishEvent xmlns:ns1="http://remote.bs.publiclayer.ws.ea.curam">
      <xmlMessage>
        <cap:alert xmlns:cap="urn:oasis:names:tc:emergency:cap:1.2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" x
              <cap:identifier>d2a42209-33a5-4fcc-8658-0fc2c08c95da</cap:identifier>
              <cap:sender>IBMCuramSample</cap:sender>
              <cap:sent>2012-07-12T15:11:14+01:00</cap:sent>
              <cap:status>Actual</cap:status>
              <cap:msgType>Alert</cap:msgType>
              <cap:scope>Public</cap:scope>
              <cap:code>KPI</cap:code>
              <cap:info>
                <cap:category>Other</cap:category>
                <cap:event>ea.ipa.pd</cap:event>
                <cap:urgency>Unknown</cap:urgency>
                <cap:severity>Moderate</cap:severity>
                <cap:certainty>Observed</cap:certainty>
                <cap:headline>Sample CAP Event</cap:headline>
                <cap:description>Sample CAP Event Description</cap:description>
                <cap:onset>2012-07-12T15:11:14+01:00</cap:onset>
                <cap:senderName>IBMCuramSample</cap:senderName>
                <cap:parameter>
                  <cap:valueName>DisposalDateTime</cap:valueName>
                  <cap:value>2012-07-12 15:11:14</cap:value>
                </cap:parameter>
                <cap:parameter>
                  <cap:valueName>DisposalDuration</cap:valueName>
                  <cap:value>0</cap:value>
                </cap:parameter>
                <cap:parameter>
                  <cap:valueName>ProgramType</cap:valueName>
                  <cap:value>areference</cap:value>
                </cap:parameter>
                <cap:parameter>
                  <cap:valueName>DisposalResult</cap:valueName>
                  <cap:value>Deny</cap:value>
                </cap:parameter>
              </cap:info>
            </cap:alert>
          </xmlMessage>
      </ns1:publishEvent>
  </soapenv:Body>
</soapenv:Envelope>
```

Figure 7. Sample CAP Event with SOAP envelope

Create a WebSphere Message Broker Flow

This section describes how to create a Message Broker Flow within the Intelligent Operations Center WebSphere Message Broker installation in order to receive and process the CAP messages sent by the Cúram Event Adaptor. This flow will accept Cúram SOAP messages, transform these messages into the precise format required by the Intelligent Operations Center, and then forward the modified messages to the Intelligent Operations Center input queue. The steps involved in creating the flow are as follows:

1. Create a new message flow file (`.msgflow`) within WebSphere Message Broker Toolkit.
2. **SOAP Input:** Create a `SOAPInput` node to expose a SOAP endpoint to accept and process the messages sent by the Cúram Event Adaptor. The node must be configured using the `EventAdaptorService.wsdl` deployable WSDL file. This file can found in `EJBServer\components\EventAdaptor\axis\EventAdaptorService` on a development installation for Cúram.
3. Create nodes to convert the message sent by the Cúram Event Adaptor into the exact format expected by the Intelligent Operations Center. This involves removing the SOAP envelope, extracting the CAP message, and removing the extra HTTP headers. The following steps can be used to achieve this:

a. **SOAP Extract:** Create a `SOAPExtract` node to remove the SOAP envelope, allowing just the body of a SOAP message to be processed.
b. **XSL Transform:** Use an XSL transformation to extract the CAP message located within the parent `<xmlMessage>` element. For example. the following XSL transformation could be used as a basis for this:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output indent="yes"/>
  <xsl:template match="/">
   <xsl:copy-of select="//xmlMessage/node()"/>
  </xsl:template>
</xsl:stylesheet>
```

Figure 8. Sample XSL Transformation

c. **HTTP Header:** Use the `HTTPHeader` node to delete all HTTP headers from the input message.
d. **Reset Content Descriptor:** Use a `ResetContentDescriptor` node to request that the message is reparsed using the XMLNSC parser.

4. **MQ Output:** Finally, the transformed message needs to be delivered to the Intelligent Operations Center `IOC.CAP.IN` MQ queue. Use an `MQOutput` node to deliver it to the `IOC.CAP.IN` MQ queue.

After creating the message flow, a BAR file should be generated. This BAR file should then be deployed to the Intelligent Operations Center WebSphere Message Broker instance.

Chapter 4. Administering the Event Adaptor

4.1 Overview

This chapter gives an overview of how to configure the Event Adaptor for use with an external system.

4.2 Enabling Event Adaptor

About this task

To send events to the Intelligent Operations Center using the Event Adaptor, it is first necessary to enable the Event Adaptor component. This is achieved via configuring a system property. To enable the Event Adaptor:

Procedure

1. Login to the application as a user with system administrative privileges and navigate to the "Property Administration" section.
2. On the "Property Administration" tab, search for "Event Adaptor Enabled Flag". This will return the "Event Adaptor Enabled Flag" property.
3. On the list row menu, select "Edit Value". This will open the "Edit Value" modal.
4. Enter "true" in the value text box and click "Save".
5. Then click "Publish..." on the tab level menu to make the changes available.

Results

The Event Adaptor is now enabled.

4.3 Enabling and Disabling External Event Types

The Event Adaptor also provides a fine grained configuration allowing administrators to enable and disable individual event types. This section describes how to achieve this.

To access the External Event Type admin page, login as a user with administrative privileges and navigate to the "Administration Workspace". Select "Event Adaptor" from the shortcuts panel. This will open the Event Adaptor tab.

Enabling External Event Types

An External Event Type will need to be enabled so that any events that use it will be active and can access the external system.

1. Select "Enable..." from the list row menu of the event to enable.
2. Click the "Yes" button on the "Enable External Event Type" modal to enable the event.

Disabling External Event Types

An External Event Type might need to be disabled because it no longer needs to have access to an external system.

1. Select "Disable..." from the list row menu of the event to disable.
2. Click the "Yes" button on the "Disable External Event Type" modal, and this will disable the event.

Enable All External Event Types

An External Event Type will need to be enabled so that any events that use it will be active and can access the external system. This offers a way to enable all events with one action.

1. To enable all events listed, select "Enable All" from the page level menu.
2. On this modal, select "Yes" to enable all events.

Disable All External Event Types

An External Event Type might need to be disabled because it no longer needs to have access to an external system. This offers a way to disable all events in one action.

1. To disable all events listed, select "Disable All" from the page level menu.
2. On this modal, select "Yes" to disable all events.

4.4 Configuring a Target System

About this task

The location of the Intelligent Operations Center WebSphere Message Broker Web Services endpoint needs to be configured so that the Event Adaptor can dispatch events to it. This is achieved through configuring the end-point URL on the Cúram Target System screen. This section describes how this is achieved. For more details on Target Systems, refer to the Target System section of the Cúram System Administration Guide.

Procedure

1. To set up a Target System, log in as "sysadmin".
2. From there, select the Target Systems shortcut (in the "Application Data" section of the shortcuts panel) to open the Target Systems tab.
3. Click "New..." to open the "New Target System" modal.
4. Enter a name for the Target System and the root URL and click "Save".
5. Click the "Add Service" from the list row menu.
6. Select the "Event Adaptor EventAdaptorService Endpoint Reference" here and enter the "Extension URL", which is the service on the exposed SOAP endpoint (refer to the WebSphere Message Broker for Cúram section for additional information on how to set up the endpoint), and, if necessary, enter the "Username" and "Password" as well.

Results

The Target System is now configured to enable the Event Adaptor to publish CAP events to the Intelligent Operations Center's instance of WebSphere Message Broker.

Chapter 5. Appendix

5.1 External Links

Intelligent Operations Center - http://www.ibm.com/software/products/us/en/intelligent-operations-center/

OASIS CAP Standard - http://docs.oasis-open.org/emergency/cap/v1.2/CAP-v1.2-os.html

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service. IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing

IBM Corporation

North Castle Drive

Armonk, NY 10504-1785

U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing

Legal and Intellectual Property Law.

IBM Japan Ltd.

19-21, Nihonbashi-Hakozakicho, Chuo-ku

Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you. Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation

Dept F6, Bldg 1

294 Route 100

Somers NY 10589-3216

U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources.

IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing

application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/us/en/copytrade.shtml.

Other names may be trademarks of their respective owners. Other company, product, and service names may be trademarks or service marks of others.

Printed in USA