CICS Transaction Server for z/OS

SupportPac CA8K - Delivering Atom feeds from CICS

CICS Transaction Server for z/OS

SupportPac CA8K - Delivering Atom feeds from CICS

Note
Before using this information and the product it supports, read the information in "Notices" on page 27.

This edition applies to Version 1.0 of SupportPac CA8K - Delivering Atom feeds from CICS and to all subsequent versions, releases, and modifications until otherwise indicated in new editions.

Contents

Preface v
What this book is about v
What you need to know to understand this book . . v

Delivering Atom feeds from CICS 1
What this SupportPac contains 1
Setting up the SupportPac files 2
What is an Atom feed? 3
Web feed terminology 3
How this SupportPac handles Atom feeds 4
Configuring the pipeline for Atom feeds 5
Specifying the terminal handler parameter list for DFH$W2FD 6
Specifying the terminal handler parameter list for DFH$W2SD 9
An example pipeline configuration file 10
Constructing a URL 12
Writing your own service routine 13
The ATOMPARAMETERS container 14
Resource Layout Mapping 17
Constant definitions 18
Sample programs and definitions 20
DFH$W2FD - the Atom feed document sample 20
DFH$W2SD - the Atom service document sample 21
DFH$W2TS - the temporary storage queue service routine 22
DFH$W2FA - a sample user-written service routine 23
Atom abends 24

Notices 27
Trademarks 28

Preface

What this book is about

This book explains how to use Atoms feeds with CICS®.

What you need to know to understand this book

This book assumes that you are familiar with CICS, either as a system administrator or as a system or application programmer.

Delivering Atom feeds from CICS

This SupportPac provides sample programs and configuration files to show you how to publish and update CICS data using the Atom publishing protocol and the Atom syndication format. This SupportPac is intended for use with CICS TS V3.1 and CICS TS V3.2 only.

This SupportPac demonstrates delivering Atom feed content from CICS temporary storage (TS) queues, or by driving a user-supplied program.

Prerequisites

To use this SupportPac, you must meet the following system requirements:

- You must have CICS APAR PK58721 installed on your system.
- You must be running z/OS version 1.7 with APAR OA16303 applied or a later release of z/OS, which contains the XML System Services parser.

What this SupportPac contains

This SupportPac contains the zipped file ca8k.zip, which includes all of the required files in several directories.

The following files are included in the SupportPac ca8k.zip file:

html/dfh$w2rx.html
This HTML file is a sample Web page that demonstrates RESTful interactions between a Web browser and a CICS resource, using the other components of this SupportPac.

html/dfh$w2fa.html
This HTML file is a sample Web page that demonstrates RESTful interactions with the CICS sample file FILEA, using the custom service routine DFH$W2FA.

images/cics-icon.gif
This graphic can be referenced as an icon in Atom feed pages.

images/cics-logo.gif
This graphic can be referenced as a logo in Atom feed pages.

license
This directory contains your license to use this SupportPac. There are thirteen translations of the license.

load/ca8k.unload
This file is an unloaded version of the load module library CA8K.LOAD that contains the z/OS executable modules DFH$W2FD, DFH$W2SD and DFH$W2TS.

load/ca8k.uncopy
This file is an unloaded version of the copy library CA8K.COPYLIB that contains copybooks and macros that are used by the sample applications.

pdf/ca8k.pdf
SupportPac documentation in PDF format.

pipe/w2atomtsqueue.xml
This XML file is a sample pipeline configuration file that you can use to configure a pipeline to manage RESTful interactions with a CICS temporary storage queue.

pipe/w2atomfilea.xml
This XML file is a sample pipeline configuration file that you can use to configure a pipeline to manage RESTful interactions with the FILEA sample file.

script/dfh$w2w2.js
This JavaScript file schedules Atom requests to CICS using an Ajax interface.

source/dfh$w2fa.cob
This program source is for the DFH$W2FA sample service routine that accesses the FILEA sample file.

source/dfh$w2fd.asm
This source is for the DFH$W2FD module that services Atom feed requests.

source/dfh$w2sd.asm
This source is for the DFH$W2SD module that services Atom service requests.

source/dfh$w2ts.asm
This source is for the DFH$W2TS module that handles interactions between DFH$W2FD and a temporary storage queue.

source/dfh$web2.dat
This file is the input data for the DFHCSDUP utility to define the CICS resources that are used by this SupportPac.

style/dfh$w2yl.css
This stylesheet modifies the appearance of the dfh$w2rx.html Web page.

Setting up the SupportPac files

You must correctly set up the files provided with the SupportPac to implement Atom feeds in CICS.

1. Unzip the ca8k.zip file to a suitable directory on your workstation.
2. Use the File Transfer Program (FTP) to transfer the files to the z/OS UNIX system where your CICS region runs.
   a. Transfer the files in the images and pdf directories as binary files.
   b. Transfer the files load/ca8k.unload and load/ca8k.uncopy as z/OS sequential files, issuing the following command:

      ```
      cd //userid
      binary
      quote site blksize=3120 lrecl=80 recfm=FB
      put ca8k.unload
      put ca8k.uncopy
      ```

      where *userid* is replaced with your own z/OS user ID.
   c. Transfer all other files in character format.
3. Use the TSO RECEIVE command to reload the ca8k.unload and ca8k.uncopy files:

   ```
   RECEIVE INDATASET(CA8K.UNLOAD)
   RECEIVE INDATASET(CA8K.UNCOPY)
   ```

Entering these commands reconstructs the CA8K.LOAD and CA8K.COPYLIB libraries.

4. Edit the source/dfh$web2.dat. The dfh$web2.dat file contains definitions for the SupportPac sample programs and CICS resources in the DFH$WEB2 RDO group.

   The definitions in the file contain several references to subdirectories of /u/uuuuuuuu/supportpac/ca8k/. Before using this file as input to DFHCSDUP, you should edit these instances to match the directories into which you have actually installed the SupportPac files with FTP in the steps described above.

5. Use the source/dfh$web2.dat file as input to the DFHCSDUP utility to create the required resource definitions.

What is an Atom feed?

Atom is both a protocol and an XML format for content providers to provide XML-based Web feeds of updated content. An Atom feed is a Web feed that uses the Atom protocol and format. This provision of updated content is known as syndicating a Web feed. Web users can subscribe to a feed, allowing them to see new content as soon as it is made available.

Atom is described by two IETF (Internet Engineering Task Force) Request for Comments documents (known as RFCs). Consult these documents for complete and authoritative information about Atom:

RFC 4287, *The Atom Syndication Format*, available from http://tools.ietf.org/html/rfc4287.

RFC 5023, *The Atom Publishing Protocol*, available from http://tools.ietf.org/html/rfc5023.

Web feed terminology

Terminology used to describe Web feeds and associated technologies is described here.

Aggregator
: An aggregator is a Web-based application or a client application that allows the user to read updates to a collection of disparate Web feeds. This service is sometimes referred to as a news aggregator or feed aggregator.

APP
: Atom Publishing Protocol. APP is a Proposed Standard (RFC 5023) for content publishing and management.

Atom
: An XML-based format for syndicated Web content and a protocol for editing and publishing Web resources that is based on the Atom Specification Format.

Feed
: A source of information on the Web that can be obtained by a feed aggregator.

RSS
: Really Simple Syndication. A feed protocol that provides similar function to Atom.

Syndication
: The process of making Web content available using a Web feed. Syndication is a way of making news items, blog entries, and other information available to a Web audience.

How this SupportPac handles Atom feeds

The SupportPac uses the pipeline support in CICS to enable clients, such as Web browsers or Atom feed aggregators, to access Atom feeds by accessing temporary storage queues or by linking to customized service routines. You must define a number of CICS resources and create a pipeline configuration file to serve Atom responses to clients.

This SupportPac creates all three types of Atom documents:

- Atom feeds, that contain one or more Atom entries
- Atom entries, that can be included in or referenced from an Atom feed
- Atom services

There are two programs that generate Atom documents in the SupportPac; DFH$W2FD generates Atom feeds and DFH$W2SD generates Atom services. The pipeline configuration file specifies the DFH$W2FD or DFH$W2SD program as a message handler. The message handler program links to one of two CICS programs, or service routines, to access the CICS resource:

- The DFH$W2TS service routine to access a temporary storage queue and operate on the records in the queue.
- Your own service routine to access arbitrary CICS resources.

The pipeline configuration file also describes the format of the records produced by the service routine.

One pipeline configuration file maps to one CICS resource.

When a client sends a request to CICS, the URIMAP resource uses the URL to identify the appropriate pipeline to process the request. The pipeline processes the request, passing any additional parameters on the URL to the message handler program. If you configure the pipeline to access a temporary storage queue, the message handler program links to DFH$W2TS to perform a number of interactions on a CICS temporary storage queue using a RESTful API:

HTTP GET
: Retrieve one or more Atom entries from appropriate records on the temporary storage queue.

HTTP POST
: Create a new record on the temporary storage queue.

HTTP PUT
: Update an existing record on the temporary storage queue.

HTTP DELETE
: Delete a record from the temporary storage queue.

It is possible to have many records on the temporary storage queue, where each record corresponds to one Atom entry. An Atom feed that consists of many Atom entries is created by reading multiple records on the TS queue. Parameters passed on the URL allow the client to select which Atom entries to interact with on the temporary storage queue.

After a document describing a record in a TS queue is published to a feed, any subscriber to that feed can view the entire content of the TS queue record.

If you configure the pipeline to use a customized service routine of your own, it must react similarly to the four HTTP methods. Use the guidance in "Writing your own service routine" on page 13 to help you to code the routine.

Configuring the pipeline for Atom feeds

To configure the pipeline to handle Atom feeds, you must create a pipeline configuration file, define, and install a PIPELINE resource.

Before configuring the pipeline, ensure that you have performed the required setup steps for the SupportPac. This provides you with the sample resources and programs to get started.

Perform the following steps to configure a pipeline for an Atom feed:

1. Create a provider pipeline configuration file. This file is an XML file that is stored in the z/OS® UNIX® System Services file system.
   a. Define the message handler program that creates the Atom document. You must specify one of the following program names:

      DFH$W2FD
      Generates an Atom feed document response.

      DFH$W2SD
      Generates an Atom service document response.

   b. Define the format of the records on the temporary storage queue using the `<handler_parameter_list>` element. The supplied programs, DFH$W2FD and DFH$W2SD, use different formats. See "Specifying the terminal handler parameter list for DFH$W2FD" on page 6 or "Specifying the terminal handler parameter list for DFH$W2SD" on page 9 for details.
   c. Do not specify the `<cics_soap_1.1_handler>`, `<cics_soap_1.2_handler>` element, or `<apphandler>` elements. The messages that are processed in the pipeline are not SOAP messages and therefore none of these elements are required.

   The pipeline configuration file must have this structure:

```
<provider_pipeline
   xmlns="http://www.ibm.com/software/htp/cics/pipeline"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.ibm.com/software/htp/cics/pipeline/provider.xsd">
  <service>
    <terminal_handler>
      <handler>
        <program>program name</program>
        <handler_parameter_list>
          Program-specific XML
        </handler_parameter_list>
      </handler>
    </terminal_handler>
   </service>
</provider_pipeline>
```

2. Copy the pipeline configuration file to a suitable directory in z/OS UNIX.
3. Change the pipeline configuration file permissions to allow the CICS region to read the file.
4. Define and install a PIPELINE resource, or use the sample PIPELINE resource, DFH$W2Q1, that is provided in RDO group DFH$WEB2. The PIPELINE resource defines the location of the pipeline configuration file. For details on how to define the PIPELINE resource, see the *CICS Resource Definition Guide*.

5. Install the URIMAP definition, or use the sample URIMAP resource, DFH$W2U1, that is provided in RDO group DFH$WEB2. You must use a URIMAP defined with USAGE(PIPELINE) and PIPELINE(*this-pipeline-definition*). For details on how to define and install a URIMAP resource, see the *CICS Resource Definition Guide*.

Your CICS system is now configured to generate Atom feed documents containing the contents of a CICS resource.

Specifying the terminal handler parameter list for DFH$W2FD

Use the DFH$W2FD program to generate an Atom feed document response.

Perform the following steps to create the handler parameter list for DFH$W2FD:

1. Create the handler parameter list with the following format:

```
<handler_parameter_list>
  <cics:feed xmlns:cics="http://www.ibm.com/software/htp/cics/atompipe">
    <atom:feed xmlns:atom="http://www.w3.org/2005/Atom">
      ....  Atom feed metadata (described below) ....
    </atom:feed>
    <cics:resource name="resource-name"
                   type="resource-type">
        <cics:fieldnames
             id="fieldname containing atom id"
             updated="fieldname containing last-updated timestamp"
             title="fieldname containing atom title"
             subtitle="fieldname containing atom subtitle"
             summary="fieldname containg atom summary">
         </cics:fieldnames>
      <cics:layout>
        .... Resource content layout description (described below) ....
      </cics:layout>
    </cics:resource>
  </cics:feed>
</handler_parameter_list>
```

   - The <cics:feed> element contains an <atom:feed> subelement, which is substantially the same as the feed element in the http://www.w3.org/2005/Atom namespace defined by Internet RFC 4287.
   - The <atom:feed> subelement contains some additional markup to define a connection between the content element of the feed and the CICS resource that is to be published.
   - The <cics:resource> element describes the CICS TS queue to be published in the feed.

2. Complete the <atom:feed> element of the <handler_parameter_list>. The <atom:feed> element is a prototype for the feed document returned by this pipeline.
   a. You can specify the following metadata subelements, as described in the Atom Syndication Format, RFC 4287:

      <atom:author>

      <atom:category>

      <atom:contributor>

      <atom:icon>

      <atom:id>

      <atom:link>

      <atom:logo>

      <atom:published>

<atom:rights>

<atom:subtitle>

<atom:summary>

<atom:title>

You can specify up to eight <atom:contributor> elements in each <atom:entry> or <atom:feed> element.

Do not specify the <atom:updated> element. It is generated automatically with a value derived from the time that the content was updated, if that time can be determined.

The <atom:generator> element is automatically generated with content indicating that the feed is generated by CICS.

b. Specify an <atom:entry> element in the <atom:feed> element for each entry in the feed document that is returned by this pipeline. You can specify the following subelements described in the Atom Syndication Format, RFC 4287:

<atom:author>

<atom:category>

<atom:content>

<atom:contributor>

<atom:icon>

<atom:id>

<atom:link>

<atom:logo>

<atom:published>

<atom:rights>

<atom:subtitle>

<atom:summary>

<atom:title>

Do not specify an <atom:source> element, as this is ignored by the runtime processing.

3. Use the inbound URL to determine which Atom component to return.

A number of different URLs can signal the use of the particular pipeline and its associated configuration file, by using the wildcarding techniques of the URIMAP. The part of the configuration file that produces the response is located by comparing the PATH component of the inbound URL against the PATH component in the **href** attribute of <link rel="self"> elements in the configuration file.

If the inbound path matches the <link rel="self"> for the feed element, an entire feed document is returned. If the inbound path matches a <link rel="self"> in an <entry> element, only a single entry document is returned.

The href value in <link rel="self"> can be relative. This relative value is a deviation from the formal Atom specification, which requires a full URL. However, by the time the Atom document is shipped, the URL is fully qualified.

Only the path component of the href attribute is used to determine which Atom component to return. The scheme and host are ignored and can usually be omitted.

4. Specify the attributes of the <cics:resource> element. The <cics:resource> element specifies the names, types, and layout of the CICS resources published in the feed.
   a. Specify the following attributes:

      name Specifies the CICS resource name.

      type For this version of the SupportPac, this value must be tsqueue or program.

   b. Optional: Specify the <cics:fieldnames> element to identify the names of fields within the CICS resource that might have additional significance in the final Atom entry document, other than in the <atom:content>. The values of the optional attributes of this element are the names of fields defined in the <cics:layout> element:

      id The name of a field that contains the atom:id value for this Atom entry.

      updated
      The name of a field that contains the atom:updated value for this Atom entry; that is, the date and time at which the entry was last updated. The field must be defined in the <cics:layout> element with an attribute of xs:dateTime. If it is also defined with attributes of dfdl_length="8" and dfdl:decimalFormat="packed", it is assumed to be a CICS ABSTIME field and is converted accordingly.

      title The name of a field that contains the atom:title value for this Atom entry.

      subtitle
      The name of a field that contains the atom:subtitle value for this Atom entry.

      summary
      The name of a field that contains the atom:summary value for this Atom entry.

   c. Specify the layout of the CICS records in the CICS resource in a <cics:layout> element. You must describe the CICS resource using the Data File Descriptor Language (DFDL).

      DFDL is an extended XML schema language that is being developed by the DFDL working group of the Open Grid Forum.

      You must define the CICS resource as a complexType. A complexType is a sequence of elements that describes a record. Each element in the record complexType must also be described as a complexType. The names and types of the fields in the records are identified by XML schema and DFDL attributes. The following table shows the DFDL attributes that are supported.

DFDL or schema attribute	Supported values	Notes
dfdl:representation	text, binary, decimal	
dfdl:length		Length of the field in bytes. The length value is ignored with any types other than string and zoned or packed decimal.
dfdl: lengthUnits	bytes	This value is the default, so you can omit it.

DFDL or schema attribute	Supported values	Notes
dfdl:decimalType	packed, zoned	Available only if dfdl:representation="decimal"
dfdl:decimalSigned	true, false	True if a zoned decimal has a sign.
xs:type	string, boolean, byte, unsignedByte, short, unsignedShort, int, unsignedInt, long, unsignedLong, dateTime	The type from the XML schema language.
xs:fractionDigits		The number of digits after the decimal point.

5. Update the <atom:entry> element with CICS resource information. The <atom:entry> element must contain unique markup that specifies how the CICS content is to be inserted into the feed element.
   a. Add cics:resource and cics:type attributes to the <atom:content> element in the <atom:entry>. These two attributes identify a <cics:resource> element that occurs subsequently in the configuration file.

   For example, the following entry element selects a record in **QUEUEA** for transmission in the feed:

```
<entry>
  <content type="text" cics:resource="QUEUEA" cics:type="tsqueue"/>
</entry>
```

   The CICS TS queue is defined subsequently as:

```
<cics:resource name="QUEUEA" type="tsqueue">
```

Your pipeline configuration file now contains a complete handler parameter list.

Specifying the terminal handler parameter list for DFH$W2SD

The DFH$W2SD program generates an Atom service document response.

Perform the following steps to create the handler parameter list for DFH$W2SD:

1. Create the handler parameter list with the following format:

```
<handler_parameter_list>
  <cics:service xmlns:cics="http://www.ibm.com/software/htp/cics/atompipe">
    <app:service xmlns:atom="http://www.w3.org/2007/app">
      .... Prototype Atom service metadata (described below) ....
    </app:service>
  </cics:service>
</handler_parameter_list>
```

   The <app:service> element is a prototype for the service document that is returned by this pipeline. You can specify in it any of the subelements that are described in the Atom Publishing Protocol, RFC 5023:

   <app:workspace>

   <app:collection>

   <app:accept>

   <app:categories>

   <atom:title>

2. Create an app:service document.

   The app:service document must have the following structure:

```
<service xmlns="http://www.w3.org/2007/app"
         xmlns:atom="http://www.w3.org/2005/Atom">
  <workspace>
    <atom:title>Title of this workspace</atom:title>
    <collection href="URL of this collection">
      <atom:title>Title of this collection</atom:title>
      <categories href="URL of a category document"/>
      <accept>Acceptable mediatype for POSTing to this collection</accept>
    </collection>
  </workspace>
</service>
```

- The service document can contain multiple `<workspace>` elements.
- Each `<workspace>` element can contain multiple `<collection>` elements.
- Each `<collection>` element must have an **href** attribute, which specifies the URL that identifies the collection. This URL is used as the target of POST requests that add new content to the collection. It can also return an Atom feed document in response to a GET request.
- Each `<collection>` element can contain multiple `<accept>` elements, which specify the acceptable mediatypes of POST requests that create new members of the collection.

Your pipeline configuration file now contains a complete handler parameter list.

An example pipeline configuration file

The following XML shows a typical pipeline configuration file for CICS Atom feed support.

```
<?xml version="1.0" encoding="UTF-8"?>
<provider_pipeline
   xmlns="http://www.ibm.com/software/htp/cics/pipeline"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.ibm.com/software/htp/cics/pipeline/provider.xsd">
 <service>
 <terminal_handler>
 <handler>
  <!-- Name of the CICS program that implements the pipeline end-point -->
  <program>DFH$W2FD</program>
  <!-- Contents of the following element go into DFH-HANDLERPLIST,     -->
  <!-- which is processed by the DFH$W2FD handler program.             -->
  <handler_parameter_list>
   <cics:feed xmlns:cics="http://www.ibm.com/software/htp/cics/atompipe">
    <atom:feed xmlns:atom="http://www.w3.org/2005/Atom">
     <!-- Unique identifier for this feed, expressed as a tag URI      -->
     <atom:id>
tag:www.ibm.com,2008-01-24:software:htp:cics:dev:00001
     </atom:id>
     <!-- Identify the path part of the URL that locates this          -->
     <!-- feed definition. This element is used by DFH$W2FD to map     -->
     <!-- the inbound URL into this feed.                              -->
     <atom:link rel="self" href="/atom/q/atomictest/feed"/>
     <atom:title>ATOM feed delivered from a CICS TSqueue</atom:title>
     <atom:subtitle>
This feed is delivered from the TSqueue ATOMICTEST in CICS.
Brought to you by CICS Transaction Server.
     </atom:subtitle>
     <!-- Optionally, insert URLs for an icon and/or logo for your -->
     <!-- feed into the following elements.                       -->
     <atom:icon>
     </atom:icon>
     <atom:logo>
     </atom:logo>
     <atom:category term="text">
     Software Development, Transaction Management
```

```
     </atom:category>
     <atom:author>
       <atom:name>Development</atom:name>
       <atom:uri>http://www.ibm.com/software/htp/cics/</atom:uri>
       <atom:email>dev@ibm.com</atom:email>
     </atom:author>
     <atom:contributor>
       <atom:name>John Doe</atom:name>
       <atom:email>jdoe@ibm.com</atom:email>
     </atom:contributor>
     <atom:rights type="text">
     </atom:rights>
     <!-- The following element and its subelements define          -->
     <!-- a single entry within the feed.                           -->
     <atom:entry>
     <!-- Unique identifier for this entry, expressed as a tag URI. -->
     <atom:id>
tag:www.ibm.com,2008-01-24:software:htp:cics:dev:00002
     </atom:id>
     <!-- Identify the path part of the URL that locates this entry -->
     <!-- definition. This element is used by DFH$W2FD to map the   -->
     <!-- inbound URL into this entry.                              -->
     <atom:link rel="self" href="/atom/q/atomictest"/>
     <atom:title>
This is an entry within the feed brought to you by
CICS Transaction Server.
     </atom:title>
     <!-- Optionally, insert a URL for an icon for your entry       -->
     <!-- into the following element.                               -->
     <atom:icon>
     </atom:icon>
     <atom:author>
       <atom:name>John Doe</atom:name>
       <atom:email>jdoe@ibm.com</atom:email>
     </atom:author>
     <atom:contributor>
       <atom:name>John Doe</atom:name>
     </atom:contributor>
     <atom:published>
     2008-01-24T12:30:00Z
     </atom:published>
     <atom:source></atom:source>
     <atom:summary>
A variety of xs:datatypes stored on a CICS TS queue.
     </atom:summary>
     <atom:rights type="text">
     </atom:rights>
     <!-- Identify the CICS resource to be incorporated into the content of  -->
     <!-- the atom entry by specifying its resource name and type. These are -->
     <!-- actually a forward reference to the cics:resource element below.   -->
     <atom:content cics:resource="ATOMICTEST" cics:type="tsqueue"/>
     </atom:entry>
   </atom:feed>
   <!-- The following is the description of the TSqueue that is being        -->
   <!-- published in the feed. It is identified by being specified in the    -->
   <!-- atom:content element above.                                          -->
   <cics:resource name="ATOMICTEST" type="tsqueue">
    <cics:layout>
    <xs:schema
      elementFormDefault="qualified" attributeFormDefault="unqualified"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:dfdl="http://dataformat.org/dfdl-1.0">
     <xs:annotation>
      <xs:documentation>
      This schema describes the layout of the CICS ATOMICTEST queue.
      </xs:documentation>
      <xs:appinfo source="http://dataformat.org/dfdl-1.0">
```

```
      <dfdl:definitions>
        <dfdl:use configuration="textTypes"/>
        <dfdl:dataFormat representation="text" encoding="ebcdic-cp-us"/>
      </dfdl:definitions>
     </xs:appinfo>
    </xs:annotation>
    <xs:element name="ATOMICTEST" cics:type="tsqueue">
     <!-- A TSqueue is a complex type that consists of a number of records. -->
     <xs:complexType>
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
       <!-- The queue record is a complex type that consists of             -->
       <!-- a sequence of fields.                                           -->
       <xs:element name="queue-record" cics:type="record">
        <xs:complexType>
         <xs:sequence>
           <!-- These DFDL definitions are for the ATOMICTEST queue.        -->
           <xs:element name="recname" type="xs:string"       dfdl:length="8" />
           <xs:element name="comm1"   type="xs:string"       dfdl:length="16"/>
           <xs:element name="int1"    type="xs:int"          dfdl:length="4" />
           <xs:element name="comm2"   type="xs:string"       dfdl:length="16"/>
           <xs:element name="short1"  type="xs:short"        dfdl:length="2" />
           <xs:element name="comm3"   type="xs:string"       dfdl:length="16"/>
           <xs:element name="ubyte1"  type="xs:unsignedByte" dfdl:length="1" />
           <xs:element name="comm4"   type="xs:string"       dfdl:length="16"/>
           <xs:element name="sbyte1"  type="xs:byte"         dfdl:length="1" />
           <xs:element name="comm5"   type="xs:string"       dfdl:length="16"/>
           <xs:element name="packed1" type="xs:decimal"      dfdl:length="16"
                                      xs:fractionDigits="2"
                                      dfdl:decimalFormat="packed"
                                      dfdl:decimalSigned="false"             />
           <xs:element name="comm6"   type="xs:string"       dfdl:length="16"/>
           <xs:element name="packed2" type="xs:decimal"      dfdl:length="16"
                                      xs:fractionDigits="3"
                                      dfdl:decimalFormat="packed"
                                      dfdl:decimalSigned="true"              />
           <xs:element name="comm7"   type="xs:string"       dfdl:length="16"/>
           <xs:element name="uint1"   type="xs:unsignedInt"  dfdl:length="4" />
           <xs:element name="comm8"   type="xs:string"       dfdl:length="16"/>
           <xs:element name="uint2"   type="xs:unsignedInt"  dfdl:length="4" />
           <xs:element name="comm9"   type="xs:string"       dfdl:length="16"/>
         </xs:sequence>
        </xs:complexType>
       </xs:element>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
   </xs:schema>
   </cics:layout>
  </cics:resource>
  </cics:feed>
 </handler_parameter_list>
</handler>
</terminal_handler>
</service>
</provider_pipeline>
```

Constructing a URL

The URL from the client can include additional parameters in a query string that select an item number from the temporary storage queue to return as an Atom entry.

The URIMAP does not use the query string to identify the pipeline and it not used by the <link rel="self" href="*xxx*"> declaration to locate the Atom feed definition. Instead it is used to locate the correct Atom document on the temporary storage queue.

1. To select specific entries from the temporary storage queue, specify the **s=** parameter as a selector in the URL's query string. If the query returns a feed document rather than an entry document, the feed document is composed of a number of entries. The actual number of entries per feed is determined by the window attribute of the <cics:feed> element in the pipeline configuration file. For example:

   ```
   <cics:feed xmlns:cics="http://www.ibm.com/software/htp/cics/atompipe" cics:window="8">
   ```

   selects 8 entries per feed. The selector value on the URL for a feed document selects the first entry in the list of entries.
2. To override the value of the window attribute in the pipeline configuration file, specify the **w=** parameter in the URL's query string.

   For example, if you specify the URL http://feed-url?s=17&w=10, the query returns a feed containing ten entries, beginning with the entry corresponding to record 17 on the temporary storage queue. The entries are returned to the client in reverse chronological order, as required by RFC 5023, so the entries appear in the sequence 17 through 8. If you do not specify a window attribute or the **w=** parameter, the window size defaults to 5.
3. To change the content type that is returned to the client, specify the **t=** parameter in the URL's query string. By default, the content element within each Atom entry document is returned in XML format. You can specify any of the following values:
   - **t=text** specifies that the returned content is in plain text without markup.
   - **t=html** specifies that the content is in escaped HTML, where the markup characters < and > are escaped to their corresponding entities < and >, as required by RFC 4287.
   - **t=xhtml** specifies that the returned content is in XHTML, where the markup characters are not escaped.

The feed document generated by CICS contains the following elements, which provide links to related feed documents containing related lists of entries:

```
<link rel="next" href="http://feed-url?s=nn"/>
<link rel="previous" href="http://feed-url?s=nn"/>
<link rel="first" href="http://feed-url?s=nn"/>
<link rel="last" href="http://feed-url?s=nn"/>
```

where the href attribute values refer to the URLs of the related feed documents, and can be used to navigate through the whole temporary storage queue record in small sections. The **t=** and **w=** parameters can be propagated as well:

```
<link rel="next" href="http://feed-url?s=nn&amp;t=xhtml&amp;w=nn"/>
<link rel="previous" href="http://feed-url?s=nn&amp;t=xhtml&amp;w=nn"/>
<link rel="first" href="http://feed-url?s=nn&amp;t=xhtml&amp;w=nn"/>
<link rel="last" href="http://feed-url?s=nn&amp;t=xhtml&amp;w=nn"/>
```

Writing your own service routine

You can write your own CICS program to provide content to an Atom feed, the program acts as a service routine. The service routine uses a number of containers and parameters to service the request.

To use your own program:

- Specify `cics:type="program"` in the `atom:content` and `cics:resource` elements.
- Specify the CICS program `name` attribute of the `cics:resource` element.

CICS will link from `DFH$W2FD` to your program with information in the **DFHREQUEST** and **ATOMPARAMETERS** containers. The **DFHREQUEST** container is present only for **POST** and **PUT HTTP** requests, and contains the **HTTP** request body.

After analyzing the request, the service routine must return a character container named `ATOMCONTENT`, containing an entire `atom:content` element for insertion into the `atom:entry` currently being processed.

Optionally, `ATOMTITLE`, `ATOMSUBTITLE`, and `ATOMSUMMARY` containers can be returned. These containers contain the content to be inserted into the `atom:title`, `atom:subtitle`, and `atom:summary` elements, respectively. The presence of these containers must be signalled by setting flags in the **ATMP_OPTIONS** parameter.

The ATOMPARAMETERS container

This container describes the parameters used to communicate with the service routine.

The `DFH$W2Px` series of copybooks map the parameters passed in the `ATOMPARAMETERS` container from `DFH$W2FD` to the resource service routine. The following copybooks are defined:

DFH$W2PD for Assembler

DFH0W2PO for Cobol

DFH$W2PL for PL/I

DFH$W2PH for C

Each parameter passed in this container is a pointer to an 8 byte area.

The first parameter is a pointer to a 64-bit options string. The options string is mapped by the **ATMP_OPTIONS_BITS** dsect.

The second parameter is a pointer to two fullwords in which the response and reason code can be returned.

The remaining parameters are pointers to pointer+length structures, in which the first word contains a pointer to the parameter's value and the second word contains its length as a binary number.

Parameters

ATMP_OPTIONS
Address of a double word containing 64 option bits. The first word is used to send options to the service routine, and the second word is used to receive options from the service routine.

ATMP_RESPONSE
Address of a double word in which the response and reason code can be returned. These are both initialized to zero, indicating successful completion.

ATMP_RESNAME
Address of a double word containing a pointer to the CICS resource name, followed by its length.

ATMP_RESTYPE
Address of a double word containing a pointer to the CICS resource type name in uppercase, followed by its length. The type can be **PROGRAM** or **TSQUEUE** .

ATMP_ATOMTYPE
Address of a double word containing a pointer to the type of Atom document being processed, in lowercase, followed by its length. The value of the type string is either ″entry″ or ″feed″.

ATMP_ATOMID
Address of a double word containing a pointer to the unique Atom request identifier, from the `atom:id` element, followed by its length.

ATMP_SELECTOR
Address of a double word containing a pointer to the selector value from the URL, followed by its length. This parameter is used to select the record in the CICS resource that is to be accessed. In the implementation shown in the SupportPac, the selector is the operand of the `"s="` keyword within the querystring section of the URL.

ATMP_HTTPMETH
Address of a double word containing a pointer to the the HTTP method padded, followed by its length. It is one of GET, POST, PUT or DELETE.

ATMP_RLM
Address of a double word containing a pointer to the Resource Layout Mapping area, followed by its length.

ATMP_MTYPEIN
Address of a double word containing a pointer to the the mediatype of the incoming HTTP request body, if any, followed by its length. It is only meaningful if the HTTP method is POST or PUT, otherwise the pointer and length are both zero.

ATMP_MTYPEOUT
Address of a double word containing a pointer to an area in which the routine must return the mediatype of the data being returned in the ATOMCONTENT container, followed by the length of that area (56 bytes). On entry to the service routine, this area contains the requested content type: ″text″, ″html″, ″xhtml″ or a mediatype such as ″text/xml″, that can be used to control the format of document returned.

ATMP_UPDATED
Address of a double word containing a pointer to an area in which the routine must return the date and time at which the returned document was last updated, followed by the length of that area (32 bytes). The value must be returned in `xs:dateTime` format, which is the same as RFC3339 format, namely yyyy-mm-ddThh:mm:ss.fffZ, or as spaces. The .fff fractional seconds are optional, and may be omitted. If spaces are returned, the current time is assumed.

ATMP_ETAGVAL
Address of a double word containing a pointer to the Etag value for the selected record, followed by its length. The Etag (or entity tag) is any string that can be used to identify the record instance uniquely.

ATMP_WINSIZE
Address of a double word containing a pointer to the feed window size, followed by its length. The value is a numeric string that contains the default number of entries to be returned in each feed.

ATMP_NEXTSEL
Address of a double word into which the service routine should set a pointer and length of a selector value for the next record in the resource, if any.

ATMP_PREVSEL
Address of a double word into which the service routine should set a pointer and length of a selector value for the previous record in the resource, if any.

ATMP_FIRSTSEL
Address of a double word into which the service routine should set a pointer and length of a selector value for the first (newest) record in the resource, if any.

ATMP_LASTSEL
Address of a double word into which the service routine should set a pointer and length of a selector value for the last (oldest) record in the resource, if any.

ATMP_ID_FLD
Address of a double word containing a pointer to the name of the field within the resource that contains the atom identifier from the `atom:id` element, if present, followed by its length. If it is present, the service routine should use this named field to store the contents of the `atom:id` element.

ATMP_UPDATED_FLD
Address of a double word containing a pointer to the name of the field within the resource that contains the time when the resource was last updated, if present, followed by its length. If no such field exists, the pointer and length are both zero. If it is present, the service routine should use this named field to locate the value of the timestamp that can be used to construct the value returned in the UPDATED parameter. This parameter may be all spaces if the resource does not contain such a field.

ATMP_KEY_FLD
Address of a double word containing a pointer to the name of the field within the resource that contains the key (RIDFLD) for File Control operations, if any, followed by its length. The parameter is not used in this version of the SupportPac.

ATMP_TITLE_FLD
Address of a double word containing a pointer to the name of the field within the resource that contains the Atom title of the represented entry, if present, followed by its length. If it is present, the service routine should use this named field to locate the entry title, and return it in the **ATOMTITLE** container.

ATMP_SUBTITLE_FLD
Address of a double word containing a pointer to the NAME OF THE FIELD within the resource that contains the Atom subtitle of the represented entry, if present, followed by its length. If it is present, the service routine should use this named field to locate the entry subtitle, and return it in the **ATOMSUBTITLE** container.

ATMP_SUMMARY_FLD
Address of a double word containing a pointer to the NAME OF THE FIELD within the resource that contains the Atom summary of the represented entry, if present, followed by its length. If it is present, the

service routine should use this named field to locate the entry summary, and return it in the **ATOMSUMMARY** container.

ATMP_OPTIONS

The options bitmap pointed to by **ATMP_OPTIONS** is mapped in two ways:

ATMP_OPTIONS_BITS and **ATMP_OPTIONS_WORDS** .

ATMP_OPTIONS_BITS is series of byte and bit definitions for use in languages that understand bit values. The bit values that have meaning are in byte **ATMP_OUTOPT_BYTE1** and are as follows:

OPTTITLE
A character string to be used as the atom:title for the entry is returned in the ATOMTITLE container.

OPTSUBTI
A character string to be used as the atom:subtitle for the entry is returned in the ATOMSUBTITLE container.

OPTSUMMA
A character string to be used as the atom:summary for the entry is returned in the ATOMSUMMARY container.

ATMP_OPTIONS_WORDS is a pair of fullword values, for use in COBOL, where bit values cannot be easily coded. The 2 fullword values are:

ATMP_OPTIONS_IN
A fullword of input option values (not used)

ATMP_OPTIONS_OUT
A fullword in which to store output option values. The fullword values equivalent to the bit values in ATMP_OUTOPT_BYTE1 are specified in Copybooks describing constant definitions, below. These values can be added together, as required, to produce a suitable bitmap value.

Resource Layout Mapping

The Resource Layout Mapping structure (RLM) is pointed to by **ATMP_RLM** in the **ATOMPARAMETERS** container.

The DFH$W2Lx series of copybooks map the contents of the Resource Layout Mapping structure (RLM):

DFH$W2LD for Assembler

DFH0W2LO for Cobol

DFH$W2LL for PL/I

DFH$W2LH for C

The Resource Layout Mapping contains a header section followed by a number of field descriptor entries.

Header

DFHRLM_HEADER
The origin of the RLM header section

RLM_EYE_CATCHER
An eight byte character string set to ″>DFHRLM<″

RLM_VERSION_MAJOR
A fullword binary number set to the major version number of the RLM

RLM_VERSION_MINOR
A fullword binary number set to the minor version number of the RLM

RLM_LENGTH
A fullword binary number set to the total length of the RLM structure

RLM_NAME
A 32-byte character string containing the name of this RLM.

RLM_STRUCT_SIZE
A fullword binary number set to the size that the entire structure represented by this RLM may occupy in storage, if this can be calculated.

Field descriptor entries

Field descriptor entries follow the header section. The entries contain the following fields, as well as some unnamed fields reserved for future use:

DFHRLM_DATA_ENTRY
The origin of the field descriptor entry.

RLM1_ENTRY_TYPE
A one-byte binary value of X'01' that identifies this descriptor as a data entry.

RLM1_CONVERT_TYPE
A one-byte binary value that represents the conversion to be applied to this field. The conversion code values are described in the DFH$W2Cx series of copybooks.

RLM1_DATA_COUNT
A halfword binary value that contains the length of the field in bytes, or, for decimal fields only, the number of decimal digits in the integer part of the value: that is, the number of digits before the decimal point.

RLM1_DATA_FRACT
A one-byte binary value that contains, for decimal fields only, the number of digits in the decimal fraction: that is, the number of digits after the decimal point.

RLM1_LOCAL_NAME_LEN
A half-word binary value that contains the length of the local name of the field.

RLM1_LOCAL_NAME_PTR
A full-word pointer that contain the address of the local name of the field.

Constant definitions

Constant values referenced by the DFH$W2Px and DFH$W2Lx copybooks are contained in the DFH$W2Cx series of copybooks.

The following copybooks are defined:

DFH$W2CD for Assembler

DFH0W2CO for Cobol

DFH$W2CL for PL/I

DFH$W2CH for C

The copybooks contain the following constants:

Binary integers representing the value of the bits in `ATMP_OUTOPT_BYTE1` , for use in `ATMP_OPTIONS_OUT` . These values are for use in COBOL programs which cannot use bit values.

OPTTITLE_NUM equivalent to OPTTITLE

OPTSUBTI_NUM equivalent to OPTSUBTI

OPTSUMMA_NUM equivalent to OPTSUMMA

The following values are used in **RLM1_CONVERT_TYPE** to represent the conversion that must be applied to convert the XML character representation of the field into its internal binary form:

CTYP_CHAR_STRING
An arbitrary character string

CTYP_BYTE
A signed binary byte value (from -128 to +127)

CTYP_UNSIGNED_BYTE
An unsigned binary byte value (from 0 to 255)

CTYP_SHORT
A signed binary halfword value (from -32768 to +32767)

CTYP_UNSIGNED_SHORT
An unsigned binary halfword value (from 0 to 65535)

CTYP_INT
A signed binary fullword integer value (from -2147483648 to 2147483647)

CTYP_UNSIGNED_INT
An unsigned binary fullword integer value (from 0 to 4294967295)

CTYP_LONG
A signed binary doubleword integer value (from -9223372036854775808 to 9223372036854775807)

CTYP_UNSIGNED_LONG
An unsigned binary doubleword integer value (from 0 to 18446744073709551615)

CTYP_BOOLEAN
A single byte boolean representation: X'80' is true and X'00' is false.

CTYP_PACKED_DECIMAL
A signed packed decimal value.

CTYP_UNSIGNED_DECIMAL
An unsigned packed decimal value.

CTYP_ZONED_DECIMAL
A signed zoned decimal value.

CTYP_UNSIGNED_ZONED
An unsigned zoned decimal value.

CTYP_CICS_ABSTIME
A timestamp in CICS abstime format. The value is the number of milliseconds since 1 Jan 1900 in the local timezone, expressed as a packed decimal field of 8 bytes.

CTYP_TOD_CLOCK
A timestamp in System z TOD clock format.

Sample programs and definitions

This SupportPac provides sample programs to demonstrate the use of Atom feeds in CICS. The DFH$WEB2 RDO group contains sample resource definitions to help you quickly deploy and run the SupportPac.

DFH$W2FD
Defines the sample feed document program.

DFH$W2SD
Defines the sample service document program.

DFH$W2TS
Defines the sample temporary storage queue service program.

DFH$W2Q1
Defines a sample PIPELINE resource for serving a temporary storage queue as a feed.

DFH$W2U1
Defines a sample URIMAP resource for serving a temporary storage queue as a feed.

WEB2HTML
Defines a sample URIMAP resource for delivering static HTML files from the html directory. This resource allows you to view the supplied HTML files from a Web browser.

WEB2IMAG
Defines a sample URIMAP resource for delivering static image files (.gif extension) from the images directory. This resource allows the cics-logo.gif and cics-icon.gif to be referenced within the Atom feed documents.

WEB2JSCR
Defines a sample URIMAP resource for the delivery of static JavaScript files from the script directory. This resource allows the dfh$w2w2.js JavaScript file to be embedded and run from the supplied HTML files.

WEB2PIPE
Defines a sample URIMAP resource for the delivery of static XML files from the pipe directory. This resource references the pipeline configuration files so that you can view them from a Web browser.

WEB2STYL
Defines a sample URIMAP resource for the delivery of static style sheets from the style directory. This resource allows dfh$w2rx.html to reference the dfh$w2yl.css style sheet.

DFH$W2FD - the Atom feed document sample

DFH$W2FD runs as a terminal handler program in a provider mode pipeline. DFH$W2FD is specified in the `<program>` element within the `<provider_pipeline>` element of the pipeline configuration file. You can use the contents of the `<handler_parameter_list>` element to configure the behavior of DFH$W2FD.

DFH$W2FD is coded in High Level Assembler. The sample program DFH$W2FD performs the following functions:

1. Checks that CICS is running in a system that supports the z/OS XML System Services parser; otherwise, DFH$W2FD ends with abend code W2P0.
2. Tries to initialize the system parser. If initialization fails, DFH$W2FD ends with abend code W2P1.
3. Passes the contents of the DFH-HANDLERPLIST container, which holds the contents of the `<handler_parameter_list>` element, to the system parser. If parsing fails, DFH$W2FD ends with abend code W2P2.
4. Analyzes the results returned by the system parser, which are a tokenized representation of the `<handler_parameter_list>`.
   a. DFH$W2FD saves all the elements of the prototype Atom feed metadata for later return in the final document.
   b. DFH$W2FD saves the attributes of the **cics:resource** element for later resolution of the CICS resource to be returned in the feed.
   c. DFH$W2FD saves the contents of the **cics:layout** element to be passed to the resource service routine.
5. When the analysis is complete, matches the path on the inbound URL against the **href** attributes in any **atom:link rel="self"** elements. The inbound URL caused the pipeline to be scheduled. The **href** attributes determine the type of document to be returned: either an Atom feed, or an Atom entry. If an Atom entry is to be returned, the **href** attributes determine which entry is returned.
6. For each **atom:entry** in the selected result document, DFH$W2FD uses the CICS resource identified by the **cics:resource** attribute of the **atom:content** element to locate a corresponding **cics:resource** element.
7. Depending on the type of CICS resource detected, an EXEC CICS LINK is made to an appropriate resource service routine. For this version of the SupportPac, the following CICS resources are supported:

   tsqueue
   a CICS tsqueue type of resource, for which the resource service routine is the sample program DFH$W2TS.

   program
   for which the resource service routine is a user-written program.
8. DFH$W2TS returns its result in the ATOMCONTENT container.
9. Uses the prototype Atom metadata that was extracted from the **atom:feed** element to construct the final result, and the contents of the ATOMCONTENT container are merged into the **atom:content** element in the **atom:entry** element of the final document.
10. Returns the final Atom document in the DFHRESPONSE container, for eventual delivery by the pipeline.
11. Sets the media type of the returned document by returning the value **application/atom+xml** in the DFHMEDIATYPE container.

DFH$W2SD - the Atom service document sample

DFH$W2SD runs as a PIPELINE terminal handler program. You specify DFH$W2SD in the **program** element of the **provider_pipeline** element of the pipeline configuration file. You can use the contents of the **handler_parameter_list** element to configure the behavior of DFH$W2SD. Its purpose is to return an Atom service document conforming to the Atom Publishing Protocol of RFC 5023.

DFH$W2SD is coded in High Level Assembler.

The sample program DFH$W2SD performs the following functions:

1. Checks that CICS is running in a system that supports the z/OS XML System Services parser; otherwise, DFH$W2SD ends with abend code W2P0.
2. Tries to initialize the system parser. If initialization fails, DFH$W2SD ends with abend code W2P1.
3. Passes the contents of the DFH-HANDLERPLIST container, which holds the contents of the **handler_parameter_list** element, to the system parser. If parsing fails, DFH$W2SD ends with abend code W2P2.
4. Analyzes the results returned by the system parser, which are a tokenized representation of the **handler_parameter_list**. DFH$W2SD saves all the elements of the prototype Atom service metadata for later return in the final document.
5. Produces the saved metadata elements, transforming any URL references into references to the current CICS server, by prefixing the current scheme and hostname to the pathname that it contructs from the metadata.
6. Returns the final Atom document in the DFHRESPONSE container, for eventual delivery by the pipeline.
7. Sets the media type of the returned document by returning the value **application/atomsvc+xml** in the DFHMEDIATYPE container.

DFH$W2SD extracts the Atom service document from the **handler_parameter_list** and changes any URL references to be references to the current CICS server. Because most of the content of an Atom service document is static, you might prefer to return the service document as simple static data, by defining it as such in a URIMAP. In this case, you must declare the mediatype of the document as **application/atomsvc+xml** in the URIMAP.

DFH$W2TS - the temporary storage queue service routine

DFH$W2FD links to DFH$W2TS if the feed document will contain content from a temporary storage queue.

DFH$W2TS is coded in High Level Assembler.

DFH$W2TS is passed the name of the temporary storage (TS) queue to be handled, and the address of a Resource Layout Mapping structure, which is a representation of the **cics:layout** element from the pipeline configuration file. DFH$W2TS is also passed the HTTP method from the original request, and the "selector" value, which is the operand of the **s=** keyword from the querystring on the originating URL.

The sample program DFH$W2TS performs functions depending on the request type:

1. For a GET request, DFH$W2TS performs the following functions:
   a. Converts the selector value to a binary halfword, and uses it as the item number in a `READQ TS` command to read an item from the selected TS queue.
   b. If the item is not found, or consists of only a single byte of X'FF', DFH$W2TS returns with a not-found return code.
   c. If the item is found, DFH$W2TS interprets it using the Resource Layout Mapping structure and converts it to an XML response, which is saved in the ATOMCONTENT container.
2. For a DELETE request, DFH$W2TS performs the following functions:
   a. Identifies the item in the same way as a GET request, and, if found, it is replaced by a single X'FF' byte.

3. For a POST request, DFH$W2TS performs the following functions:
   a. Initializes the z/OS XML System Services parser in the same way as DFH$W2FD.
   b. Obtains the DFHREQUEST container and passes it into the parser.
   c. Analyzes the parser result. For each XML element found, its content is converted using the Resource Layout Mapping structure into the appropriate representation in a field of the TS queue.
   d. If all the conversions are complete, writes the composite result to the TS queue.
   e. Sets an HTTP status of 201 into the DFHHTTPSTATUS container to show that a new entry has been created.
   f. Processes the GET method to return the new TS queue record as a formatted Atom entry in the ATOMCONTENT container.
4. For a PUT request, DFH$W2TS processing is similar to POST, except that an existing TS queue record is updated instead of a new one being written.

DFH$W2FA - a sample user-written service routine

This sample program demonstrates writing your own customized service routine. The program provides a RESTful interface to the CICS sample file, FILEA. It is not a general purpose file handler service routine.

Before using DFH$W2FA you must create the FILEA VSAM file and install an appropriate RDO definition for it. An appropriate RDO definition is provided in the CICS-supplied sample RDO group DFH$FILA. These steps are described in the CICS Installation Guide. You must also ensure that the FILEA file is enabled and open.

The DFH$W2FA sample program performs the following steps:

- Obtains the invocation parameters by accessing the ATOMPARAMETERS container.
- Obtains the selector parameter that was derived from the `s=` parameter in the querystring. This selector parameter will be used as the key for accessing the requested FILEA record. If the key is not present, the key of the first record on the file is obtained. The key must be exactly six digits. DFH$W2FA does not pad the key value to the correct length.
- Saves the Etag value from the If-Match header, if an Etag value was provided.
- Obtains the HTTP method for the request. It uses the HTTP method name to determine the subsequent action:
  - For a GET request, reads the record implied by the selector key value, and returns it contents.
  - For a PUT request, DFH$W2FA reads the record implied by the selector key value, using a read for update function. It calculates the Etag value for the record just read and compares it with the one that was received from the If-Match header. If these are not equal the file is unlocked from the read for update, and a 412 HTTP response is set in the DFHHTTPSTATUS container. Otherwise the data provided in the DFHREQUEST container is parsed and used to update the file record. The file record is rewritten with a REWRITE function.
  - For a POST request, the data provided in the DFHREQUEST container is parsed and used to write a new file record, using the key value derived from

the selector. If the write fails because of a duplicate key condition, an HTTP response of 409 is returned. If the write is successful, an HTTP 201 response is returned.

– For a DELETE request, the record with the key specified in the selector is deleted.

The data in the DFHREQUEST container is parsed using the COBOL XML PARSE function. The IN-CONTENT flag is used to keep track of whether the XML element being analyzed is within the atom:content element, or not. When the IN-CONTENT flag is set to Y, the content of elements whose names are recognized are saved in the FILEA record structure. All other elements are ignored. Note that rigorous analysis of XML namespace prefixes is not attempted. The `atom:content` element must use ″atom″ as its namespace prefix, and the fieldname elements must be unprefixed.

For all requests except DELETE, the file record contents are returned by the RETURN-FILE-CONTENT procedure, which performs the following steps:

- Embeds the fields of the FILEA record in an XML structure. This XML structure is saved into the ATOMCONTENT container.
- Creates a title for the Atom entry and saves it in the ATOMTITLE container. A flag indicating this is set in the ATMP-OPTIONS-OUT parameter.
- Creates a summary for the Atom entry and saves it in the ATOMSUMMARY container. A flag indicating this is set in the ATMP-OPTIONS-OUT parameter.
- Calculates a checksum of the new file record and converts it to a hexadecimal value. This hexadecimal value is returned in the ATMP-ETAGVAL parameter for use as an Etag value.
- Calls the ADD-NAVIGATION-LINKS procedure to return the key values of the first, last, next, and previous file records. These key values will be used for creating navigation link URLs if the entry is embedded within a feed document.

Atom abends

If an error occurs during the processing of a request for an Atom feed, the SupportPac issues an abend.

W2C0 Unsupported conversion. When converting a data field from an XML character string to the corresponding binary representation in the CICS resource, or vice versa, an unsupported conversion was attempted.

W2C1 Invalid data for conversion. When converting a data field from an XML character string to the corresponding binary representation in the CICS resource, or vice versa, the data provided was invalid for the attempted conversion.

W2E0 Required Content-type absent. For an HTTP POST or PUT request the required Content-Type header was omitted, so the data type of the request body cannot be determined.

W2E1 Unknown data type specified in the configuration file. The type specified in the `xs:type` or `dfdl` attributes of the `xs:element` element is not recognized or not supported.

W2E2 CICS resource not in configuration. A CICS resource specified in the `cics:resource` attribute of an `<atom:content>` element could not be located in a `<cics:resource>` element in the configuration file.

W2E3 I/O error accessing resource. The CICS resource service routine detected an I/O error when attempting to access the CICS resource.

W2E4 Invalid RLM entry type. An invalid entry was detected in the Resource Layout Mapping structure passed to the CICS resource service routine.

W2E5 Unrecognized HTTP method. An HTTP method other than GET, POST, PUT, or DELETE was detected.

W2E6 Special field invalid type. The type specified for one of the fields named in the `<cics:fieldnames>` element was incompatible with its intended usage as Atom metadata.

W2E7 XML stack overflow. The nesting level of XML elements has exceeded the size of the stack used to record the element structure.

W2P0 System XML Parser not available. An attempt was made to run this SupportPac on a z/OS system that does not contain the required z/OS XML System Services parser.

W2P1 Parser initialization failed. The GXL1INI parser initialization routine failed when initializing the XML parser for parsing the pipeline configuration file.

W2P2 Parser execution failed. The GXL1PRS parsing function failed when parsing the pipeline configuration file.

W2P3 Parser initialization failed. The GXL1INI parser initialization routine failed when initializing the XML parser for parsing an incoming request body.

W2P4 Parser execution failed. The GXL1PRS parsing function failed when parsing an incoming request body.

Notices

The provisions set out in the following two paragraphs do not apply in 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.

Information contained and techniques described in this publication have not been submitted to any formal IBM test and are distributed on an ″AS IS″ basis.

The use or implementation of any information contained and/or of any technique described in this document is the user's responsibility and depends on the user's ability to evaluate and integrate the information and/or technique into the user's operational environment. While IBM has reviewed each item for accuracy in a specific situation, IBM offers no guarantee or warranty that the same or similar results will be obtained elsewhere. Users attempting to adapt any technique described in this document to their own environments do so at their own risk.

The information contained in this publication could include technical inaccuracies or typographical errors.

Changes are periodically made to the information contained 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 reference in this publication to an IBM licensed program or another IBM product is not intended to state or imply that only IBM's program or other product may be used. Any functionally equivalent program that does not infringe applicable intellectual property rights may be used instead of the referenced IBM licensed program or other IBM product.

The user is responsible for evaluating and verifying the operation of the material supplied in conjunction with this publication in conjunction with other products, except those expressly designated by IBM.

International Business Machines Corporation may have patents or pending patent applications covering subject-matter described in this document. The furnishing of this document does not give you any license to any such patent. You can send license inquiries, in writing, to:

The IBM Director of Licensing
International Business Machines Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

CICS	IBM	MQSeries
RACF	z/OS	

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

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