



## **Deploy CICS data Atom feeds into your mashups in minutes**

*By Peter Havercaan, CICS Development, IBM Hursley*

---

## Contents

---

2	<b><i>Abstract</i></b>
2	<b><i>Rich Internet applications and mashups</i></b>
3	<b><i>Situational applications</i></b>
3	<b><i>The Atom protocols</i></b>
4	<b><i>Atom and the REST architectural style</i></b>
4	<b><i>The CICS contribution</i></b>
5	<b><i>The Atomservice resource</i></b>
7	<b><i>CICS Atom support as a REST service</i></b>
7	<b><i>Rapid deployment of business applications</i></b>
11	<b><i>Defining and installing the Atomservice resource</i></b>
11	<b><i>Checking that the Atomservice behaves as an Atom feed</i></b>
12	<b><i>Constructing the RIA or mashup</i></b>
13	<b><i>Using CICS-supplied samples</i></b>
14	<b><i>Using the IBM Mashup Center</i></b>
17	<b><i>Using Enterprise Generation Language (EGL)</i></b>
19	<b><i>Summary</i></b>
19	<b><i>For more information</i></b>

## Abstract

An enormous amount of business data is currently managed by CICS® applications. One of the aims of IBM CICS Transaction Server for z/OS® Version 4.1 is to unlock that data and to make it available in modern rich Internet applications (RIAs) without requiring extensive CICS application programming skills. CICS now makes some of its internal business assets available as Atom feeds, which can be consumed by standard Web client toolkits and presented to end users, either alone or in combination with data from other servers, to produce composite mashup applications. When the CICS data is combined with data from other sources, it can easily be used to produce new and highly intuitive displays of that data, giving the viewer an informed view of aspects of the business represented by the CICS data. Because CICS uses the Atom Syndication Format to produce its output, the data can also be simply consumed as a feed to which end users can subscribe using most modern Web browsers or dedicated feed readers.

## Rich Internet applications and mashups

The explosive availability of the Internet and the World Wide Web has led to a new breed of applications that run within Web browsers, but provide many of the performance and usability characters of traditional desktop applications. These are known as rich Internet applications, or RIAs, which exploit the dynamic document-handling capabilities of modern Web browsers. The dynamic nature of these applications is achieved by executing scripts, typically in the JavaScript™ programming language, which interact directly with the layout engine of the browser. A mashup is a specific kind of RIA in which inputs from multiple servers are merged together in a composite Web page to produce a novel result.

---

## Highlights

---

***The Internet Engineering Task Force has introduced two new formal standards for feeds under the generic name Atom—the Atom Syndication Format and the Atom Publishing Protocol.***

### **Situational applications**

Situational applications are rapidly deployed simple applications developed by a small team or a single developer to meet a specific short-term need. They are not necessarily RIAs, but the RIA toolkits are sufficiently mature that developing an RIA or mashup is now often synonymous with a situational application.

### **The Atom protocols**

The ability to serve content as a list of frequently changing items is the principle behind a “feed,” also known as a syndication. The Internet Engineering Task Force has introduced two new formal standards for feeds under the generic name Atom—the Atom Syndication Format<sup>1</sup> and the Atom Publishing Protocol.<sup>2</sup> The Atom Syndication Format is primarily about delivering read-only feeds, but the Atom Publishing Protocol (informally known as AtomPub) effectively describes how to edit or update the feed content. When a feed is made editable by the AtomPub protocol, it is described as a collection.

An Atom feed or collection document is actually a composite structure of multiple entry documents. Each feed or collection document is identified by a URL, but each of the individual entries within it is also identified by its own specific URL. Entry documents can therefore be referenced independently of the feed document that contains them.

It is important to notice that, although the terminology talks of publishing a feed, the feed protocols do not actually “push” data towards the feed client. In fact, feed readers periodically send “pull” requests (actually HTTP GET requests) to the feed server. Therefore, the responsiveness of a feed reader to changes in the feed depends entirely on the frequency with which these requests are sent.

---

**Highlights**

---

**Atom and the REST architectural style**

AtomPub collections can be manipulated over HTTP using the concepts of Representational State Transfer (the REST architectural style). This means that the four standard HTTP methods can be sent to an Atom collection, or an entry within a collection, with the following results:

HTTP method	Atom document type	Action
GET	Entry	Entry document is returned.
PUT	Entry	Existing entry is modified.
DELETE	Entry	Specified entry is removed from the collection.
GET	Collection	Collection document is returned (wholly or partially).
POST	Collection	New entry is added to the collection.

***CICS Transaction Server 4.1 allows certain CICS assets to be published as feeds or manipulated as AtomPub collections, allowing developers of RIAs and situational applications to incorporate CICS data into their applications.***

**The CICS contribution**

The contribution of CICS Transaction Server 4.1 to the new technologies is to allow certain CICS assets to be published as feeds or manipulated as AtomPub collections. This allows developers of RIAs and situational applications to incorporate CICS data into their applications.

CICS files and temporary storage queues can be published automatically without additional programming. In addition, CICS application programs can be nominated as the targets of the Atom requests. With some additional programming, these programs can be used to deliver content from other types of CICS-managed assets such as Web Services or databases. The support for publishing temporary storage queues means that it is now extremely simple to publish CICS data to an Atom feed—all the application needs to do is to write the data to a queue.

CICS Transaction Server for z/OS V4.1 also contains support for systems management over a REST-style interface. This allows the properties of a wide variety of CICS resources to be manipulated, and is the basis for the new IBM CICS Explorer™.

### **The Atomservice resource**

In order to publish a CICS asset as an Atom feed, CICS uses a new resource called an Atomservice. The Atomservice describes some of the Atom attributes of the feed to be published, and also specifies the name and type of the CICS asset whose content is to be published. The Atomservice is associated with an inbound HTTP request by means of a URIMAP resource. The URIMAP has a new USAGE attribute value of ATOM, and a new attribute of ATOMSERVICE.

### **The Atomservice configuration file**

The Atom metadata that is used to construct an Atom response is quite complex, and it is inappropriate to specify it with traditional CICS RDO attributes, so an auxiliary file in the IBM z/OS UNIX® file system is used to specify additional Atom characteristics. These characteristics are encoded in XML.

### **The Atomservice XML binding file**

The raw data in a CICS file or temporary storage queue is just a sequence of bytes, and it is normally a CICS application program that interprets those bytes to give them meaning. When the contents of the resource are published in a feed, the raw data bytes have to be transformed into meaningful text strings that can be inserted into the XML feed documents. The binding file describes the mapping between the raw data bytes and an equivalent representation of those bytes in an XML structure.

**How CICS delivers an Atom feed**

When CICS receives an inbound HTTP request whose request URL matches a URIMAP with USAGE(ATOM), control is passed to an Atom request handler, which uses information in the Atomservice resource to process the request. The request handler uses Atomservice attributes from the configuration file to decide what sort of Atom document is to be returned, and which CICS resource is to be used to populate the response. If the CICS resource is a file or a temporary storage queue, CICS uses the XML binding file to transform the raw data bytes contained in the resource into the equivalent textual form that is suitable for transmission in the Atom response document. Figure 1 shows the interactions between the request URL and the CICS resources to which it relates.

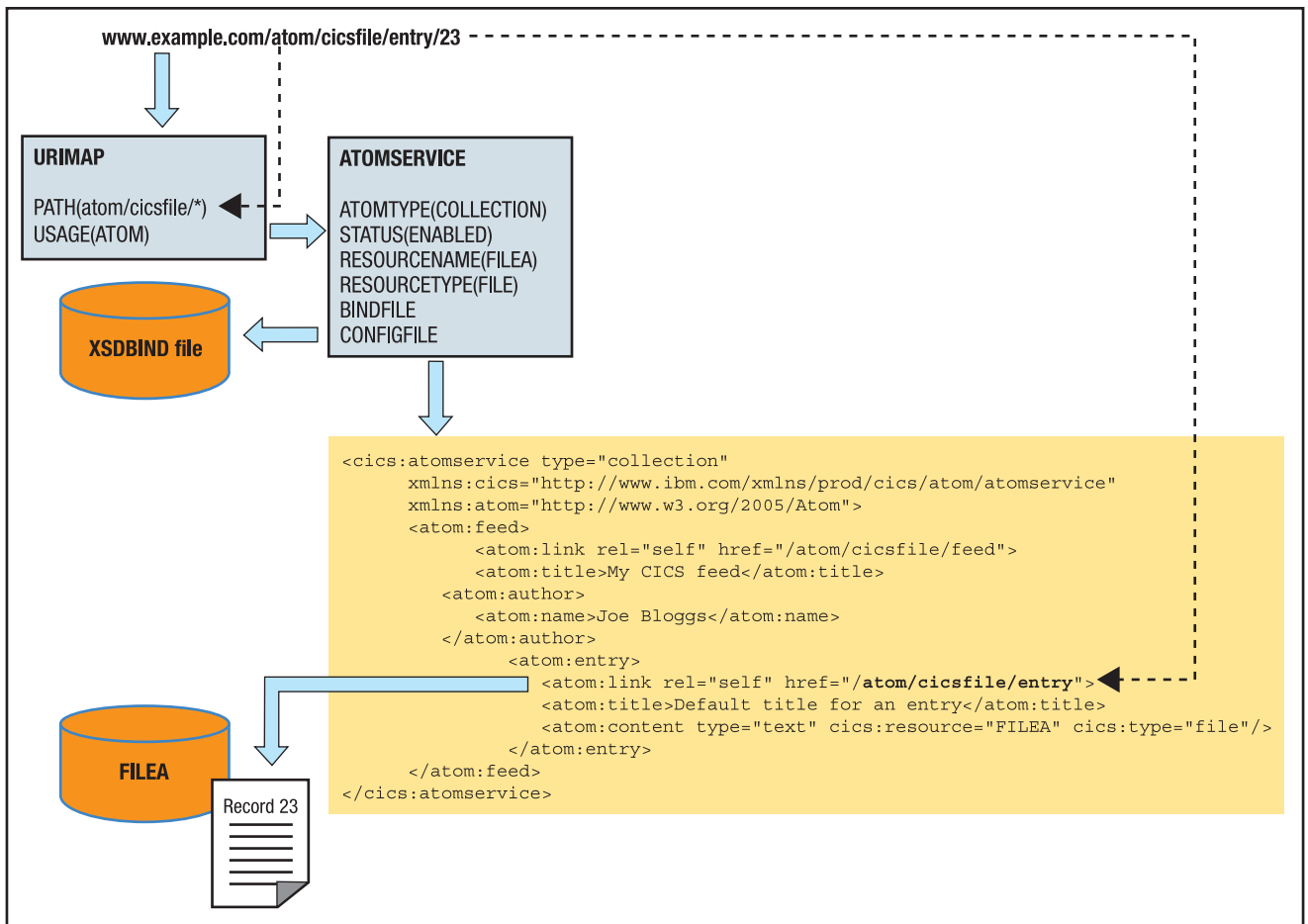


Figure 1: Mapping a URL request to a file record

### **Security for Atom feeds**

Because the CICS Atom feed processing is based on the existing CICS Web Support, the underlying CICS security facilities are available. You nominate the TCP/IP port for the feed with a TCPIPSERVICE definition. Here you can also specify the security associated with that port. The SSL attribute specifies whether to use Secure Sockets Layer to encrypt the feed data, and the AUTHENTICATE attribute specifies whether the user should be prompted for a username and password, or for an SSL client certificate.

When you use these authentication options, the feed request executes under the nominated user identity, and access controls for that user identity are applied. You can control whether the authenticated user has access to the particular Atomservice, or to the CICS resource being published as the feed.

### **CICS Atom support as a REST service**

When the Atomservice describes the Atom response document type as a feed, only the HTTP GET method is supported for it. CICS returns an Atom feed document or an Atom entry document, depending on the format of the request URL. But when the Atomservice describes the Atom response document type as a collection, a wider variety of HTTP methods are supported, which give access to the CICS resource in a REST style, as described earlier.

### **Rapid deployment of business applications**

To build a situational application or RIA using CICS data, we have to assume that CICS already contains relevant business data that can be used in the RIA. Assuming that the data is available in a CICS VSAM file, the first step is to create an XML binding file that describes the layout of the business data within the file records.

### Creating the XML binding file

The most important artifact used in the transformation of the CICS application data is the XML binding file described earlier. There are a number of ways of creating this using the CICS-supplied XML Assistant, which is a pair of offline batch utilities. The simplest technique is to use the DFHLS2SC program to convert a language structure into an XML schema file, which also creates the relevant XML bind file. In this context, a language structure is just a description of the layout of the file records in a high-level language such as COBOL, PL/I, or C. If the file is already being used in a CICS application, such a language structure almost certainly exists in the application development environment.

Not all language structures are suitable for input to the XML Assistant. If the file record structure is overly complex, it may not be possible to create an XML binding file that represents it. The constraints are described in the “High-level language and XML schema mapping” topic in the CICS Transaction Server for z/OS V4.1 Information Center.<sup>3</sup> In particular, the COBOL clauses OCCURS DEPENDING ON, OCCURS INDEXED BY, and REDEFINES are not supported. More complex structures may require the use of IBM Rational® Developer for System z®—the CICS Web service integrated development environment—to generate driver applications.<sup>4</sup> If no language structure is available, or one cannot be reconstructed, it may be possible to construct an XML binding file using the DFHSC2LS program, which uses an XML schema file or a WSDL file (Web Services Description Language) as input.



### Creating the Atomservice configuration file

The next required input for creating the CICS Atom feed is the Atomservice configuration file. Before you create this, you need to decide the following:

- *What URL path will be used to access the entire feed? This is the URL that will be used by your RIA to download multiple entries within the feed using the GET method, or to add new entries to the feed using the POST method. This is specified as /atom/cicsfile/feed in Figure 1. It is referred to below as the feed URL path.*
- *What URL paths will be used to access individual members of the feed? These are the URLs that will be used by your RIA to download single entries within the feed using the GET method, or to update or delete single entries within the feed using the PUT or DELETE methods. In the configuration file, you specify a single generic prototype for these URLs. In Figure 1, this is specified as /atom/cicsfile/entry. The URL path that you specify here should be different from the one you specified for the entire feed, but the initial parts of the paths should be the same so that you can define a single URIMAP that references both forms.*
- *How will you identify the individual member elements within the feed? This is how you qualify the URL defined in the previous bullet. The prototype URL you chose there will be extended by a “selector” value, which is the string that actually identifies each instance of a record within the CICS resource. You can choose whether the selector value is a decimal number, a hexadecimal number, or a character string.*

The Atomservice configuration file is encoded in XML and contains three major sections:

- *A section that describes the CICS resource that will be used to populate the feed.*
- *A section that contains the Atom metadata for the whole feed. This is where you specify the path of the URL for the feed, in the href attribute for the <link rel="self"/> element.*
- *A section that contains the Atom metadata for the individual entries within the feed.*

You can write the configuration file from scratch using documentation in the CICS Information Center, or use the sample `/usr/lpp/cicsts/cicsts41/samples/web2.0/atom/filea.xml` as a prototype. Figure 2 is a copy of the configuration file reduced to its minimal elements.

```
<?xml version="1.0"?>
<atomservice type="collection"
  xmlns="http://www.ibm.com/xmlns/prod/cics/atom/atomservice">
  <feed>
    <resource name="FILEA" type="file">
      <bind root="DFH0CFIL"/>
    </resource>
    <authority name="example.com" date="2009-02-14"/>
  </feed>
  <feed xmlns="http://www.w3.org/2005/Atom">
    <link rel="self" href="/atom/f/filea/feed"/>
    <title>Sample CICS file FILEA</title>
    <subtitle>A RESTFUL service for FILEA</subtitle>
    <author><name>CICS Development</name></author>
    <entry>
      <link rel="self" href="/atom/f/filea"/>
      <title>FILEA item</title>
      <author><name>CICS Development</name></author>
      <rights>Copyright (c) Example Corp 2009</rights>
      <published>2009-02-28T12:00:00Z</published>
      <content xmlns:cics="http://www.ibm.com/xmlns/prod/cics/atom/atomservice"
        cics:resource="FILEA" cics:type="file"/>
    </entry>
  </feed>
</atomservice>
```

Figure 2: Writing the Atomservice configuration file

### **Defining and installing the Atomservice resource**

Once you have defined the configuration file and the XML binding file, you can specify them in an Atomservice definition that also specifies the name and type of the CICS file that you are publishing as an Atom Pub collection. You should then also define a URIMAP in which the PATH attribute is the common part of the two URLs that you previously specified for the feed and the entries. The URIMAP should specify USAGE(ATOM) and also specify the name of the Atomservice you just defined in its ATOMSERVICE attribute. Now install both the Atomservice and the URIMAP, together with a TCPIP SERVICE, to define a port upon which CICS should listen for the Atom feed requests.

### **Checking that the Atomservice behaves as an Atom feed**

Once you have successfully installed the Atomservice, the URIMAP, and the TCPIP SERVICE, you should already be able to view the CICS file as a feed. In a Web browser, enter the URL host and port implied by the TCPIP SERVICE with the feed URL path described above, for instance: <http://www.example.com/atom/cicsfile/feed>

If everything is behaving correctly, your browser should show a page with a feed display containing the first few entries from your file. However, the actual content of your file records will probably not be displayed. This is because the content is expressed in XML, and most browsers and feed readers do not attempt to format this when it is delivered in a feed. The content should nevertheless be present, and you should be able to see it through the “View Source” option of your browser.

---

### Highlights

---

***Once you have confirmed that the CICS Atom support is publishing your data as a feed, you can begin to construct the RIA that will consume or edit the feed.***

### Constructing the RIA or mashup

Once you have confirmed that the CICS Atom support is behaving as it should by publishing your data as a feed, you can begin to construct the RIA that will consume or edit the feed. The RIA is a JavaScript application that can send Ajax requests to perform GET, POST, PUT, and DELETE requests to the Atom URLs that you defined in your Atom service. The script must extract the XML content element from the Atom documents it receives, using appropriate XML parsing functions, and apply it to the business application in the RIA. When using the POST and PUT methods, the RIA must construct a complete Atom entry document, which is sent in the body of the request.

Developing the JavaScript application is mostly a non-CICS activity, with one caveat. The RIA will execute as a JavaScript application in your browser, and will use Ajax to send HTTP messages to CICS to perform the RESTful interactions. But for security reasons, a script that uses Ajax to communicate with a server must also originate from that same server. This is known as the Same Origin Policy control. There are two ways of achieving this—either the JavaScript files that contain the Ajax programming can be downloaded directly from CICS as static content, or a reverse proxy server can be used to hide the different origins of the JavaScript and the CICS feed data. If you deliver the JavaScript files from CICS, you need to define URIMAPs to deliver JavaScript files, with media type `application/javascript`, from a specific z/OS UNIX file system directory that is established for this purpose.

### Using CICS-supplied samples

CICS Transaction Server for z/OS V4.1 includes a very simple Ajax application that demonstrates the concepts you need. To see this in operation, simply install the DFH\$WEB2 RDO group using CEDA. Figure 3 shows typical output from this sample, which can be accessed as [http://samplehost/web2.0/html/dfh\\$w2q1.html](http://samplehost/web2.0/html/dfh$w2q1.html), where samplehost is the host name (and port, if necessary) of the CICS Transaction Server for z/OS V4.1 system. This sample uses RESTful HTTP messages to write personnel data to a CICS temporary storage queue, which can then be consumed as an Atom feed.

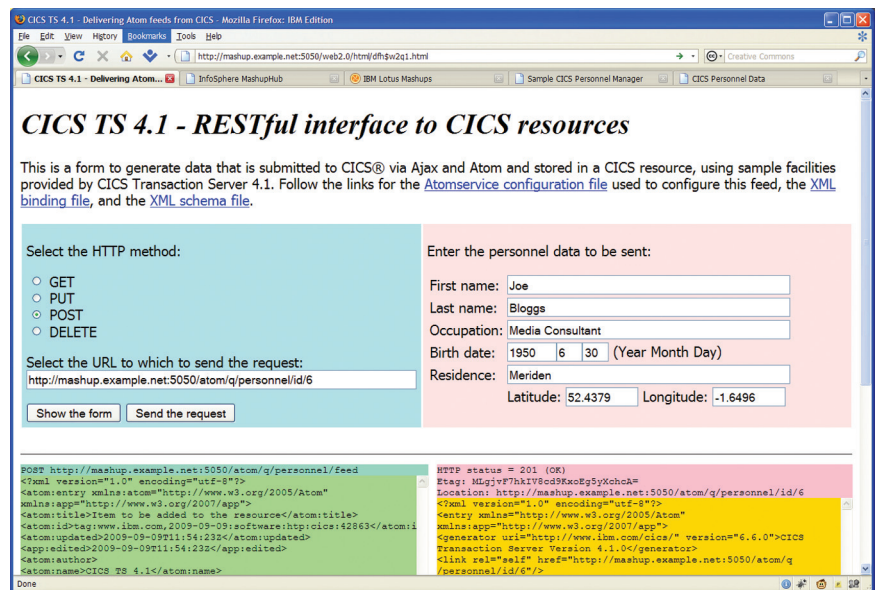


Figure 3: Personnel data sample

---

## Highlights

---

*The IBM Mashup Center provides a collection of tools to construct business mashups with little or no programming effort.*

The sample is described in full detail in the CICS Transaction Server for z/OS V4.1 Information Center. It uses a JavaScript file `dfh$w2w2.js` to perform the Ajax communication between the HTML and CICS. Because this script is delivered as static content from CICS, it meets the requirements of the Same Origin Policy.

### Using the IBM Mashup Center

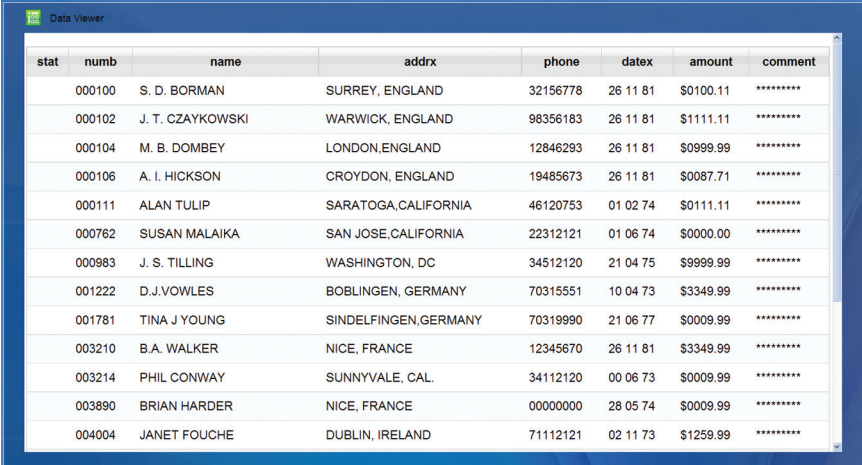
The IBM Mashup Center is a collection of tools to construct business mashups with little or no programming effort. Its two major components are the IBM InfoSphere™ MashupHub and IBM Lotus® Mashups.<sup>5</sup> These components can both be used to consume the feeds produced by CICS, and display their contents in a variety of representations using only graphical drag and drop interfaces. The InfoSphere MashupHub can be used to transform the contents of a feed into a different structure. It also provides the ability to save widgets developed across your enterprise into a catalog, which can then be used as a source for developing further mashups and RIAs in Lotus Mashups.

The InfoSphere MashupHub also acts as an Ajax proxy, which hides the fact that different feeds are sourced from different servers. This avoids problems introduced by the Same Origin Policy since—as far as the client application is concerned—all the mashup data originates from the same server.

Because CICS provides fully functional Atom feeds which are primary input components for the Mashup Center, it is very easy to deploy the CICS feed into a mashup in a matter of minutes.

### Accessing a CICS feed from the IBM Mashup Center

You can access the content of a CICS Atom feed directly from Lotus Mashups by using the Data Viewer widget. Unlike most feed readers, this widget understands XML content within a feed and displays it as a table under headings that are the same as the XML element names. So if you have installed the DFH\$WEB2 group and also set up the FILEA sample file, you can immediately display the contents of the file by entering the URL <http://samplehost/atom/f/filea/feed> in the “Edit Settings” for the Data Viewer. The results should be similar to Figure 4, which is the same data that you would see using the AMNU transaction from a 3270 transaction in CICS, but in a much more visually appealing style.



The screenshot shows a window titled "Data Viewer" containing a table with 8 columns: stat, numb, name, addrx, phone, datex, amount, and comment. The table lists 16 rows of customer data, including names, addresses, phone numbers, and transaction amounts.

stat	numb	name	addrx	phone	datex	amount	comment
000100		S. D. BORMAN	SURREY, ENGLAND	32156778	26 11 81	\$0100.11	*****
000102		J. T. CZAYKOWSKI	WARWICK, ENGLAND	98356183	26 11 81	\$1111.11	*****
000104		M. B. DOMBEY	LONDON, ENGLAND	12846293	26 11 81	\$0999.99	*****
000106		A. I. HICKSON	CROYDON, ENGLAND	19485673	26 11 81	\$0087.71	*****
000111		ALAN TULIP	SARATOGA, CALIFORNIA	46120753	01 02 74	\$0111.11	*****
000762		SUSAN MALAIKA	SAN JOSE, CALIFORNIA	22312121	01 06 74	\$0000.00	*****
000983		J. S. TILLING	WASHINGTON, DC	34512120	21 04 75	\$9999.99	*****
001222		D.J.VOWLES	BOBLINGEN, GERMANY	70315551	10 04 73	\$3349.99	*****
001781		TINA J YOUNG	SINDELFINGEN, GERMANY	70319990	21 06 77	\$0009.99	*****
003210		B.A. WALKER	NICE, FRANCE	12345670	26 11 81	\$3349.99	*****
003214		PHIL CONWAY	SUNNYVALE, CAL.	34112120	00 06 73	\$0009.99	*****
003890		BRIAN HARDER	NICE, FRANCE	00000000	28 05 74	\$0009.99	*****
004004		JANET FOUCHE	DUBLIN, IRELAND	71112121	02 11 73	\$1259.99	*****

Figure 4: View FILEA in the Lotus Mashups Data Viewer

---

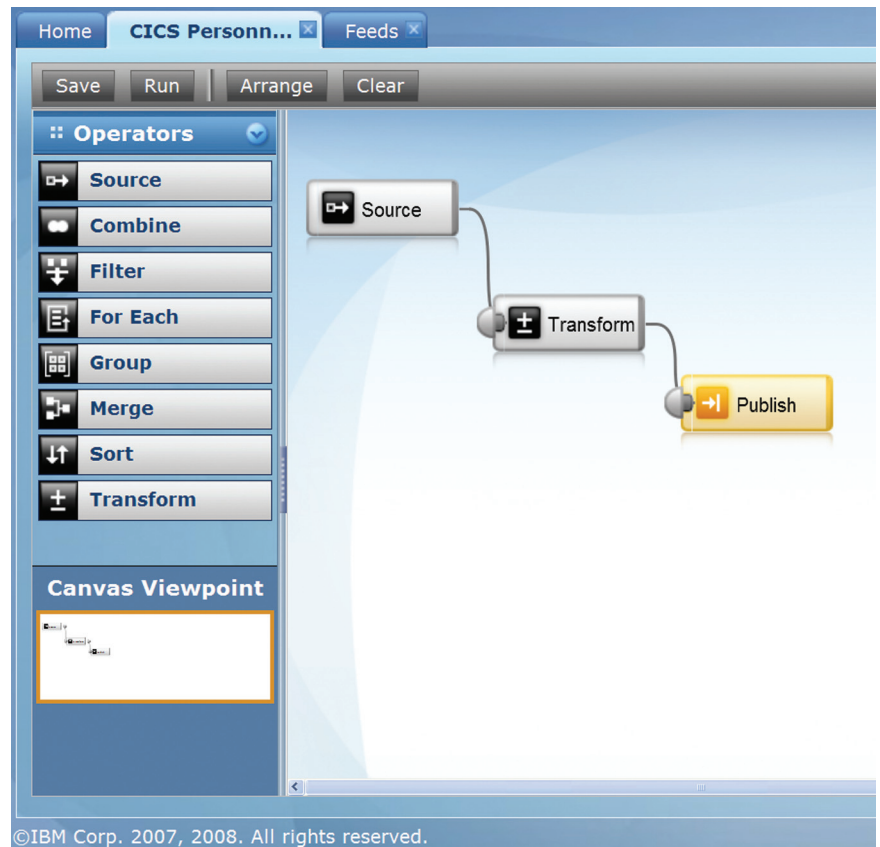
### Highlights

---

*InfoSphere MashupHub provides services to transform the contents of one feed and output them as another feed after applying transformations to the data.*

#### Using MashupHub to transform CICS feed data

Unless you designed the contents of your CICS feed very carefully, it is quite likely that the data in the feed is not in a format that is suitable for consumption by other widgets. For example, the CICS Personnel Data sample feed is designed to deliver the latitude and longitude of each person's residence, but the OpenStreetMap widget that is provided with Lotus Mashups cannot accept these as two separate values. But the InfoSphere MashupHub provides services to transform the contents of one feed and output them as another feed after applying some transformations to the data. So the latitude and longitude emitted by CICS can be transformed into a comma-separated pair of values (which must be specified with longitude before latitude), which can then be accepted by the OpenStreetMap widget of LotusMashups. As shown in Figure 5, InfoSphere MashupHub uses a graphical "wiring" representation for connecting widgets together.



©IBM Corp. 2007, 2008. All rights reserved.

Figure 5: Transforming a CICS feed in IBM InfoSphere MashupHub



Once the transformation widget above has been saved in the InfoSphere MashupHub catalog, it can then be used in Lotus Mashups (rather than the raw CICS feed). The resultant mashup takes the personnel data from CICS, transforms the person's residence location information, and displays it on the map, as shown in Figure 6.

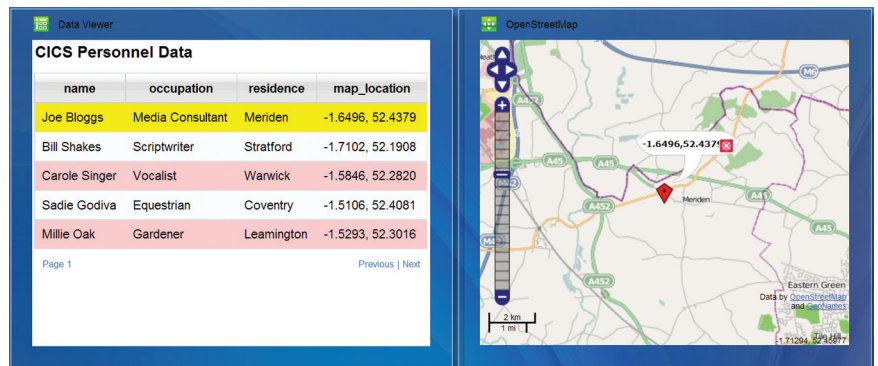


Figure 6: Transformed CICS feed data linked to OpenStreetMap

### Using Enterprise Generation Language (EGL)

While some mashup applications may be easy to assemble with a tool such as IBM Mashup Center, others may require more extensive logic and processing to be created in JavaScript. EGL is a source-code language and RIA development technology featured in IBM Rational Business Developer and in three other products, including Rational Developer for System z and EGL Community Edition, a free RIA development environment.<sup>6</sup>

---

### Highlights

---

***EGL facilitates taking corporate data or syndication feeds like Atom and making that data available to Web browsers as traditional Web-based or Web 2.0-style RIA applications.***

EGL facilitates taking corporate data or syndication feeds like Atom—as processed by long-standing, non-EGL applications—and making that data available to Web browsers as traditional Web-based or Web 2.0-style RIA applications. EGL provides two significant benefits for source-code development:

- *An easy-to-use visual development environment that provides many conveniences, especially at development time*
- *A business-focused development language to help the developer think about business issues instead of the details of relatively low-level technologies*

The EGL Rich UI provides an interactive source-code editor that can help the EGL developer visually assemble or quickly write a client-side RIA that is conceptually simple and is ultimately deployed as JavaScript.<sup>7</sup> Developers can follow a wizard-driven four-step process to access any service from the EGL Rich UI application. The service can be RESTful (such as an Atom feed from CICS) or SOAP-based (as is traditional for Web services), or services can be created for resources such as relational databases, programs (including CICS transactions and z/OS batch programs), files (including z/OS-based VSAM files), IBM WebSphere® MQ message queues, and more.

Services accessed by the EGL Rich UI can be displayed, acted upon, or manipulated using a set of included RIA widgets, similar to those in the IBM Mashup Center. Rational Business Developer also helps create unique EGL programs and widgets to process and refine the data from the services in customized ways, creating functionally complete applications which provide the responsiveness and feel of a full-fledged desktop application.

---

## Highlights

---

*The business data derived from CICS can ultimately be absorbed into modern graphical display formats swiftly, with confidence, and with little to no programming.*

### Summary

With CICS Transaction Server for z/OS V4.1 you can now rapidly expose legacy CICS data into the modern arena of RIAs and mashups, simply by installing an AtomService resource and its associated configuration files. Once the data is published as a feed, it can be consumed by a number of browser-based processes and desktop widgets.

Without any further transformation by the client, the data can be viewed as a simple feed. But with little programming effort in the browser, using the services of widely available JavaScript program libraries, the business data derived from CICS can ultimately be absorbed into modern graphical display formats. It can also be composed with data from other servers to produce novel views of the data that would be impossible within CICS itself.

Furthermore, when the CICS data is integrated into the graphical tools provided with the IBM Mashup Center or a development environment such as Rational Business Developer and EGL, the production of complex mashup visual representations can be performed swiftly, with confidence, and with little to no programming.

### For more information

To learn more about enriching your rich Internet applications with CICS data Atom feeds, or to upgrade to IBM CICS Transaction Server for z/OS V4.1, please contact your IBM sales representative or IBM Business Partner, or visit: [ibm.com/cics](http://ibm.com/cics)



© Copyright IBM Corporation 2009

IBM Corporation  
IBM Systems and Technology Group  
Route 100  
Somers, NY 10589  
U.S.A.

Produced in the United States of America  
September 2009  
All Rights Reserved

IBM, the IBM logo, ibm.com and CICS are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (@ or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at [ibm.com/legal/copytrade.shtml](http://ibm.com/legal/copytrade.shtml)

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. 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, or service names may be trademarks or service marks of others.

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates.

The information contained in this documentation is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this documentation, it is provided "as is" without warranty of any kind, express or implied. In addition, this information is based on IBM's current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this documentation or any other documentation. Nothing contained in this documentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM (or its suppliers or licensors), or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

IBM customers are responsible for ensuring their own compliance with legal requirements. It is the customer's sole responsibility to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws.

<sup>1</sup> The Atom Syndication Format.

[www.ietf.org/rfc/rfc4287.txt](http://www.ietf.org/rfc/rfc4287.txt)

<sup>2</sup> The Atom Publishing Protocol.

[www.ietf.org/rfc/rfc5023.txt](http://www.ietf.org/rfc/rfc5023.txt)

<sup>3</sup> IBM CICS Transaction Server for z/OS, Version 4 Release 1 Information Center.

<http://publib.boulder.ibm.com/infocenter/cicsts/v4r1/index.jsp?topic=/com.ibm.cics.ts.home.doc/welcomePage/WelcomePage.html>

<sup>4</sup> For more information on Rational Developer for System z, please refer to the IBM white paper entitled "Achieving business resilience through integrated systems management," September 2009, or visit <http://www-01.ibm.com/software/awdtools/rdz>

<sup>5</sup> IBM Mashup Center.

[www.ibm.com/software/info/mashup-center](http://www.ibm.com/software/info/mashup-center)

<sup>6</sup> EGL Community Edition.

<http://www-01.ibm.com/software/rational/products/eglce>

<sup>7</sup> For tutorials, case studies, and additional information on EGL, visit the EGL Café at [www.ibm.com/rational/eglcafe](http://www.ibm.com/rational/eglcafe), or refer to *Enterprise Web 2.0 with EGL* ([www.mc-store.com/5107.html](http://www.mc-store.com/5107.html))



Recyclable, please recycle

ZSW03133-USEN-00