HostBridge Interfaces

HostBridge supports a variety of ways in which another application can invoke its services. We refer to these as "connection methods." Two fundamental interfaces underlie these various connection methods: the HTTP interface and the LINK interface.

- **HTTP Interface** – This interface allows communication with HostBridge through the exchange of HTTP requests and responses over a TCP/IP network. HostBridge does not provide its own HTTP server. Instead, HostBridge relies upon one of the IBM provided HTTP servers that run on the mainframe.

- **LINK Interface** – This interface allows a local or remote program to request the services of HostBridge by "linking" to it; that is, invoking the services of HostBridge as though it were calling a subroutine. The input to, and output from, HostBridge are passed using a communication area, or "COMMAREA." The LINK interface is the foundation of a number of communication methods supported by HostBridge (e.g., MQ and TIBCO).

Table of Contents

- HTTP Interface
  - Connection Architecture
- LINK Interface
  - Connection Architecture
- Controlling XML Document Size
- EBCDIC Encoding
- Trailing Spaces in the COMMAREA

HTTP Interface

HyperText Transfer Protocol (HTTP) is the most commonly used protocol between XML-enabled applications. This section describes how to use HostBridge with HTTP connections. By supporting HTTP, remote applications can invoke the services of HostBridge through the exchange of HTTP requests and responses over a TCP/IP network. HostBridge does not provide its own HTTP server. Instead, HostBridge relies upon one of the IBM-provided HTTP servers that run on the mainframe:

- CICS HTTP Listener – An integrated HTTP Server that is part of CICS TS 1.3 or later. The HTTP Listener is a component of CICS Web Support.
- OS/390 HTTP Server (formerly Domino Go)
- WebSphere/390

An HTTP request indicates which "method" to use to request services from the remote application. The two most common request methods are GET and POST. The GET method typically retrieves information from the remote process. The POST method typically provides input data to a remote process. However, GET requests can also specify input data and parameters to the remote process using a "query string." As a result, the only practical difference between GET and POST requests is how they convey input data and parameters. HostBridge supports both GET and POST requests. When using a GET request to invoke HostBridge services, applications must format the input data and parameters as a HostBridge "command string" (discussed later). The GET request conveys the HostBridge command string as the query string of the GET request.

When using a POST request to invoke the services of HostBridge, input data and parameters can be specified as either a command string, an XML document, or as a SOAP compliant request. When you use a command string, the HTTP request header must include the following:

```
Content-Type: application/x-www-form-urlencoded
```

When you use an inbound XML document or SOAP request, the HTTP request must include the following HTTP header:

```
Content-Type: text/xml
```

Connection Architecture

The diagram below shows the basic connection architecture for using HTTP to communicate with HostBridge.
Regardless of the HTTP server you use, a typical transaction includes these steps:

1. The external application sends an HTTP request to the host.
2. The host receives the HTTP request and validates access; if access is authorized, the request is passed to HostBridge for processing.
3. HostBridge starts the transaction on the host, provides it the relevant input data, and receives the output from the transaction.
4. HostBridge converts the output data to XML and sends it back to the external application as part of the HTTP response.

**LINK Interface**

The LINK interface allows a local or remote program to request the services of HostBridge by "linking" to it; that is, invoking the services of HostBridge as though it were calling a subroutine. The input to, and output from, HostBridge passes a communication area, or "COMMAREA." The LINK interface is the foundation of a number of communication methods supported by HostBridge (e.g., MQSeries and TIBCO). You can use the LINK interface in a number of ways:

- **Direct** – A CICS program can invoke the services of HostBridge by directly LINKing to HostBridge.
- **EXCI** – A program running outside the CICS environment, but within the mainframe operating system environment, can use the External CICS Interface (EXCI) to invoke the services of HostBridge.
- **ECI** – A program running outside the mainframe environment can use the external call interface (ECI) to invoke the services of HostBridge. For example, ECI is used to invoke the services of HostBridge via IBM's CICS Transaction Gateway (CTG) or Microsoft's COM Transaction Integrator (COMTI).

**Connection Architecture**

The diagram below shows the basic connection architecture for using a COMMAREA to communicate with HostBridge.

Using the LINK interface, the requesting program passes the input to HostBridge via the COMMAREA. The COMMAREA must contain a HostBridge command string. After executing the requested transaction, HostBridge returns the resulting XML document to the requesting program via the COMMAREA.
When using the LINK interface, you should be aware of issues concerning XML document size, URL encoding, and how HostBridge handles trailing spaces in a COMMAREA.

**Controlling XML Document Size**

The maximum size of a COMMAREA that applications can use to pass data to/from a CICS program is roughly 32KB. When specifying input to HostBridge via a COMMAREA this is not usually a constraint. However, it can be a constraint when delivering the XML document generated by HostBridge to the requesting program.

The size of XML documents returned by HostBridge is directly proportional to the number and size of the fields conveyed. For transactions with large numbers of fields and/or large amounts of field data, the resulting XML document can be larger than 32KB. To reduce the size of the HostBridge XML response, include the command HB_XML_MIN=1 on the HostBridge request. This command reduces the number of elements and attributes included in the XML document returned from HostBridge. This command sets the default value for the following HostBridge commands to 0:

- HB_ATTR_BYTE
- HB_ATTR_JUSTIFY
- HB_ATTR_MDT
- HB_NAME_ATTRIB
- HB_NAME_INDEX
- HB_SHOW_FULL_PROT
- HB_VALUE_LEN
- HB_VALUE_MAXLEN

To enable any attributes or elements disabled by HB_XML_MIN, you must set the appropriate command values equal to 1. For example, HB_ATTR_BYTE=1 enables the byte attribute while HB_XML_MIN is enabled. To override HB_XML_MIN, commands must appear in a specific order. HostBridge processes commands in the order they appear, whether they are specified in a command string or an XML document. For example, a command string with HB_XML_MIN=1&HB_ATTR_BYTE=1 instructs HostBridge to include attribute byte information in the XML document; whereas HB_ATTR_BYTE=1&HB_XML_MIN=1 does not.

**EBCDIC Encoding**

The LINK interface supports %xx encoding in the HostBridge command string, where “xx” is the hex value of an EBCDIC character. Since HostBridge parses the name/value pairs of a command string using “=” and “&” as delimiters, you must encode those characters when they are used as data:

<table>
<thead>
<tr>
<th>Character</th>
<th>EBCDIC %xx Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>%7e</td>
</tr>
<tr>
<td>%</td>
<td>%50</td>
</tr>
</tbody>
</table>

For example, if HostBridge receives a name/value pair such as “name=test user”, there is no need to encode “=” because the character is an operator not data. However, if HostBridge receives a name/value pair such as “amount=number=200”, where the value is ‘number = 200’, you must encode the = character using %7e. Thus, you would encode “amount=number=200” as “amount=number%7e200”. When %7e appears within a value, HostBridge decodes the string and converts it back to ‘number=200’. You can encode spaces natively as a space character or as %40.

**Trailing Spaces in the COMMAREA**

When the requesting program uses the LINK interface and a HostBridge command string to invoke HostBridge, it is important to understand how HostBridge handles trailing spaces in the COMMAREA. HostBridge trims the blank spaces off the end of the COMMAREA until it encounters a non-space, non-NULL (binary 0) character. Thus, if the last command in the command string includes trailing spaces, you must encode the trailing spaces (or at least the last trailing space) as %40 (an EBCDIC space). The following examples illustrate how HostBridge handles trailing spaces within a COMMAREA:
Table 2-2. Trailing spaces in COMMAREA using the LINK interface

<table>
<thead>
<tr>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>'HB_ENTRY=tran</code></td>
<td>Five trailing spaces. HostBridge will drop the trailing spaces and only pass along the non-space characters.</td>
</tr>
<tr>
<td><code>'HB_ENTRY=tran%40'</code></td>
<td>Five encoded trailing spaces. HostBridge will correctly recognize the spaces and pass them to the CICS application.</td>
</tr>
<tr>
<td><code>'HB_ENTRY=tran %40'</code></td>
<td>Five trailing spaces where the last space is encoded. HostBridge will correctly recognize the spaces and pass them to the CICS application.</td>
</tr>
<tr>
<td><code>'HB_ENTRY=tran HB_DEBUG=0'</code></td>
<td>Since the five trailing spaces are not the last thing in the COMMAREA, HostBridge interprets the spaces correctly. Note that since HB_DEBUG is disabled by default, the command has no effect other than to signal the end of the string.</td>
</tr>
</tbody>
</table>