CICS Transaction Server for z/OS
5.6

CICSPlex SM CMCI REST API Reference

Note

Before using this information and the product it supports, read the information in Product Legal Notices.

This edition applies to the IBM® CICS® Transaction Server for z/OS®, Version 5 Release 6 (product number 5655-Y305655-BTA) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

About this PDF

This PDF is a reference of the CMCI REST application programming interface for the CICSPlex SM element of CICS Transaction Server for z/OS. This documentation is intended for system programmers who are writing HTTP-based system management clients to interact with CICSPlex SM.

For details of the terms and notation used in this book, see Conventions and terminology used in CICS documentation in IBM Documentation.

Date of this PDF

This PDF was created on 2024-04-22 (Year-Month-Date).

Chapter 1. CMCI REST API overview

The CICS management client interface (CMCI) provides a REST application programming interface (API) for system management clients such as IBM CICS Explorer®. The CMCI REST API can also be used in an automated process, by leveraging the Ansible IBM z/OS CICS collection.

The client initiates an HTTP request to the CMCI. If the interface determines that the request is valid, it constructs a CICSPlex® SM API command or, in the case of a stand-alone CICS region, a CICS system command. After running the command the CMCI creates an HTTP response. If the request is successful, this takes the form of an HTTP 200 (OK) response and an XML feed containing a result set, which it passes back to the client. If the request is not successful, the response consists of a non-OK HTTP response code with details of the failure.

The format for CMCI HTTP requests and responses is based on the HTTP/1.1 protocol. See The HTTP protocol for more information about this protocol.

Elements in a CMCI request

A CMCI request takes the form of an HTTP header followed by a URI (Universal Resource Identifier) and, where appropriate, an XML body containing details of any changes to be made to CICS or CICSPlex SM resources.

HTTP header

The header incorporates one of the following HTTP methods:

HTTP method	Description
DELETE	Removes resources from the CICSPlex SM data repository, removes resources from the CSD, or discards installed resources.
GET	Retrieves information about resources on the CICSPlex SM data repository, retrieves information about resources on the CSD, or retrieves information about installed resources.
POST	Creates resources on the CICSPlex SM data repository or resources in the CSD.
PUT	Updates existing resources in the CICSPlex SM data repository, updates existing resources in the CSD , or sets attributes and performs actions on installed resources. Also performs actions on CICSPlex SM and CSD resources.

URI

The URI includes the name of a CICS or CICSPlex SM resource, and specifies a series of parameters that refine the scope and nature of the query to identify one or more instances of the specified resource. In a GET request, the URI also specifies whether the API retains or discards a set of results. If the API retains the results, a new request can act on the retained results without having to repeat the retrieval operation. You can also use subsequent requests to page through the retained results selecting one or more records at a time. See “Using retained result sets to improve the performance of GET requests” on page 9 for more information about retaining result sets.

XML body

POST and PUT requests include an XML body. In a PUT request the body contains either details of the changes to be made to resource attributes, or the action to be performed on the targeted resources. In a POST request, the body incorporates the attribute values you want to give to the new resource instance.

GET and DELETE requests do not require an XML body. If additional parameters are required for a DELETE request, those parameters must be included in the URI and can optionally be added to the XML body. For example; if you are deleting a CSD resource definition you must

include PARAMETER=CSDGROUP(*csdgrp*) in the URI, and you can optionally add <parameter name="CSD"/> to the XML body.

CMCI requests

You can use the CICS management client interface (CMCI) to develop HTTP client applications that manage installed and definitional CICS and CICSPlex SM resources. A CICS management client interface request takes the form of an HTTP header followed by a URI (Universal Resource Identifier) and, where appropriate, an XML body. The four CICS management client interface requests are DELETE, GET, POST and PUT.

CMCI DELETE requests

The CICS management client interface (CMCI) uses the HTTP DELETE method to remove resources from the data repository, or to discard installed resources from CICS or CICSPlex SM.

The client forms a DELETE request using an HTTP header comprised of the following parts:

- The method name, in this case DELETE
- The URI that identifies the resources to be deleted
- The HTTP version
- Authorization credentials if required

DELETE requests can operate on either installed resources or on definitional resources. The interface constructs a CICS DISCARD command when operating on operational resources, and a CICSPlex SM REMOVE command when operating on definitional resources.

For example, to delete all transaction definitions in CICSplex PLEX1 that have a name beginning with TR and a program beginning with P, the client constructs the following HTTP header:

```
DELETE /CICSSystemManagement/CICSDefinitionTransaction/PLEX1?CRITERIA=NAME%3DTR%2A%20
AND%20PROGRAM%3DP%2A HTTP/1.1
Host: example.com:23792
Authorization: Basic RlJFRDpQQVNTVzBSRA==
```

The initial line has three parts, separated by spaces:

- The method name
- The request URI, which is specified as an absolute path that begins `/CICSSystemManagement/` immediately followed by the external name that identifies the resource together with the parameters and filters that determine which instances of that resource are to be deleted. To make the URI suitable for processing by the CICS management client interface, the client replaces certain restricted characters such as spaces and asterisks with escape sequences.
- The HTTP version, which, when connecting to the CICS management client interface, is always HTTP/1.1

The second line identifies the host name and port number of the target system separated by a colon as specified in the URI. This line ends with a carriage return, followed by a line-feed.

The optional third line contains authorization credentials. If your system is running with the CICS system initialization parameter **SEC** as YES, you must supply a user ID and password in base-64 in a basic authentication header. In this example the user ID is *FRED* and the password is *PASSWORD*.

The HTTP header is then finalized by a final carriage return and line-feed on a separate line.

Response to a DELETE request

On the completion of a DELETE request, the client receives a response from the CMCI made up of an HTTP header, and an XML feed containing a result summary and details of the deleted resources.

The header consists of an HTTP response code, the date and time, and details of the server and content type. The following example shows the header for a successful DELETE request:

```
HTTP/1.1 200 OK
Cache-Control: no-store
Date: Tue, 02 Jun 2009 14:51:37 GMT
Server: IBM_CICS_Transaction_Server (zOS)
Content-Type: application/xml; charset=UTF-8
Transfer-Encoding: chunked
```

The body of the response consists of a <response> root element containing a <resultsummary> element displaying summarized information about the request, for example:

```
<response xmlns="http://www.ibm.com/xmlns/prod/CICS/smw2int"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ibm.com/xmlns/prod/CICS/smw2int
http://example.com:30061/CICSSystemManagement/schema/CICSSystemManagement.xsd"
version="1.0" connect_version="0410">
<resultsummary api_response1="1024" api_response1_alt="OK"
api_response2="0" api_response2_alt="" recordcount="1" successcount="1" />
</response>
```

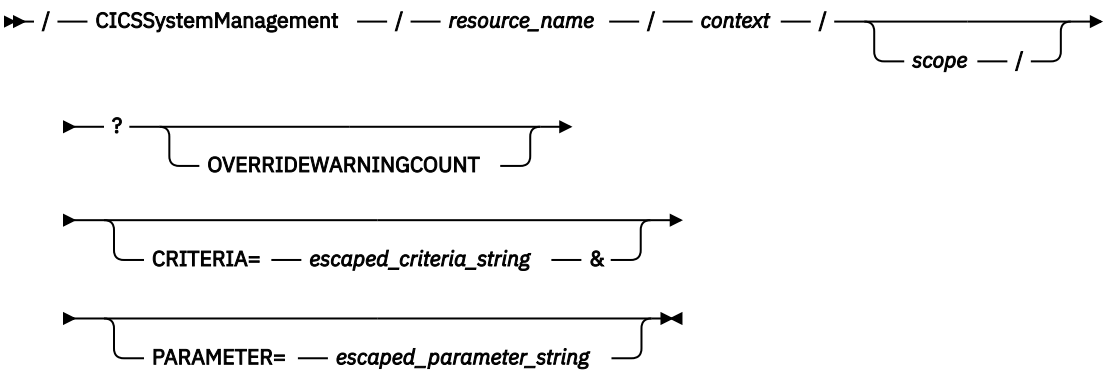
Security considerations

When you are using HTTP methods, be aware that some firewalls do not allow HTTP PUT or DELETE traffic through the firewall because of security considerations. To accommodate this restriction, you can use a CMCI POST request to tunnel a DELETE request through a POST request.

CMCI DELETE request URI

The URI of a delete request identifies the instances of the resources to be deleted by the operation.

The following diagram illustrates the URI syntax for a DELETE request.



Options

/CICSSystemManagement
Indicates that the request is accessing the CICS management client interface. CICSSystemManagement is case-sensitive.

resource_name
Specifies the external resource name that is associated with the CICS or CICSPlex SM resource that is being deleted. For example, specifying the resource name CICSLocalFile associates the request with the CICSPlex SM LOCFILE resource table, and CICSResourceAssignmentDefinition associates the request with the RASGNDEF resource table. See “CMCI resource names” on page 17 for a complete list of external resource names.

The HTTP DELETE method is not valid for all resources. To determine whether this HTTP method is valid for a particular resource, see the description of the resource in CICSPlex SM resource tables.

Resource names are not case-sensitive. If the client specifies an incorrect resource name, an HTTP 404 (Not Found) response is returned.

context
If the CMCI is installed in a CICSPlex SM environment, *context* is the name of the CICSplex or CMAS associated with the request; for example, PLEX1.

If the CMCI is installed as a single server, *context* is the application ID of the CICS region associated with the request.

The value of context must not contain spaces. Context is not case-sensitive.

scope
Specifies the name of a CICSplex, CICS group, CICS region, or logical scope associated with the request. The scope is a subset of the context and limits the request to particular CICS systems or resources. Scope is not mandatory. If it is absent, the request is limited by the value of the context alone. The scope must not contain spaces. It is not case-sensitive.

OVERRIDEWARNINGCOUNT
Bypasses the warning count limit mechanism and allows the request to execute as if the default warning count limits were not specified. This option is effective only when the CICSPlex SM Web User Interface server initialization parameter **RESOURCELIMIT** is set to WARNING.

Query parameters
Refine the scope and nature of the request. The constituent parts of the query section can occur in any order, but each can occur only once in a URI. Query parameters are separated with an ampersand (&). Although query parameter values are not case-sensitive, certain attribute values must have the correct capitalization because some attributes such as TRANID and DESC can hold mixed-case values. The query parameters are as follows:

CRITERIA=*escaped_criteria_string*
A string of logical expressions that filters the data returned on the request. The string that makes up the value of the CRITERIA parameter follows the same rules as the filter expressions in the CICSPlex SM application programming interface. For more guidance about specifying filter expressions using the CICSPlex SM API, see How to build a filter expression.

You must replace certain restricted characters with escape sequences to ensure that they can be interpreted correctly by the server. See “CMCI escape sequences” on page 41 for guidance about using escape sequences in CICS management client interface URIs.

PARAMETER=*escaped_parameter_string*
A string of one or more parameters and values of the form **parameter_name**(*data_value*) that refines the request. The rules for specifying these parameters are the same as in the CICSPlex SM application programming interface. For more guidance about specifying parameter expressions using the CICSPlex SM API, see Performing an action against a resource.

As with the criteria string, you must encode certain characters with escape characters. See “CMCI escape sequences” on page 41 for guidance about using escape sequences in CICS management client interface URIs.

CMCI GET requests

The CICS management client interface (CMCI) uses the HTTP GET method to retrieve resources from CICS or CICSPlex SM.

The client forms a GET request from the following parts:

- The HTTP method, in this case GET
- The URI that identifies the resources to be retrieved and indicates whether the results are stored for later use or discarded
- The HTTP version
- Authorization credentials if required

For example, to retrieve all CICS local files in the CICSplex PLEX1, the client might construct the following request:

```
GET /CICSSystemManagement/CICSLocalFile/PLEX1/ HTTP/1.1
Host: example.com:22958
Authorization: Basic RlJFRDpQQVNTVzBSRA==
```

The initial request line of the HTTP header has three parts, each separated by spaces and ended by a carriage return, followed by a line-feed:

- The HTTP method name
- The request URI, which is specified as an absolute path that begins `/CICSSystemManagement/` immediately followed by the external name that identifies the resource, and the parameters and filters that determine which instances of that resource are to be retrieved.

  A GET request can either operate directly on CICS or CICSplex SM resources or on results retained from a previous GET request. Normally results are discarded at the end of each request but you can retain a set of results by specifying the NODISCARD option on the GET request URI. When you use this option, the CMCI checks the status of all result sets and discards only those that have been unused for 15 minutes or more.

  If the request is operating directly on a resource, specify a resource name, which can be of an operational or a definitional resource, add the context and optionally the scope, and then further refine, sort, and filter the results by including one or more query parameters. Specify the ORDERBY parameter on the GET request URI to sort the results by one or more attributes.

  If the request operates on a set of retained results, you identify the results by replacing the resource name with `CICSResultCache` and specifying a cache token.You can narrow down the request to one or more records in the retained results by adding values for the **index** and **count** options.
- The HTTP version, which, when connecting to the CICS management client interface, is always HTTP/1.1

The second line identifies the host name and port number of the target system separated by a colon as specified in the URI. This line ends with a carriage return, followed by a line-feed.

The optional third line contains authorization credentials. If your system is running with the CICS system initialization parameter **SEC** as YES, you must supply a user ID and password in base-64 in a basic authentication header. In this example the user ID is *FRED* and the password is *PASSWORD*.

The HTTP header is then finalized by a final carriage return and line-feed on a separate line.

Response to a GET request

On the completion of a GET request, the client receives a response from the CICS management client interface made up of an HTTP header, an XML feed containing a result summary, and details of the retrieved resources. However, if the URI included the SUMMONLY parameter, the response consists only of the HTTP header and the result summary.

The header consists of an HTTP response, the date and time, and details of the server and content type. The following example shows the header for a successful GET request:

```
HTTP/1.1 200 OK
Cache-Control: no-store
Date: Wed, 06 Aug 2008 08:32:00 GMT
Server: IBM_CICS_Transaction_Server (zOS)
Content-Type: application/xml; charset=UTF-8
Transfer-Encoding: chunked
```

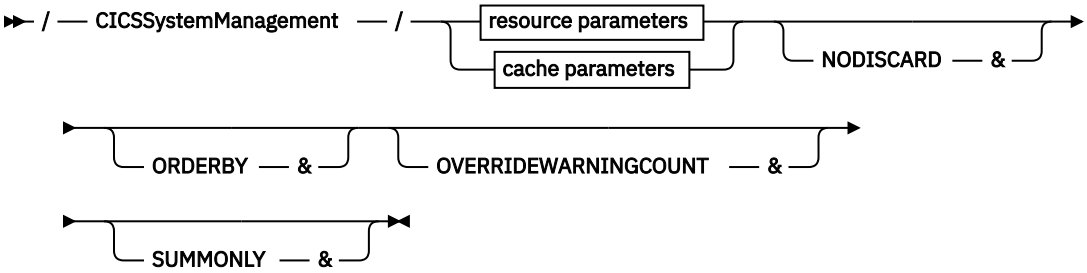
The body of the response consists of a <response> root element containing a <resultsummary> element displaying summarized information about the request, and typically a <records> element containing details of the retrieved resources. You can suppress the <records> element of the response by specifying the **SUMMONLY** parameter in the URI. If the request completed with errors, diagnostic information is provided in an <errors>element.

CMCI GET request URI

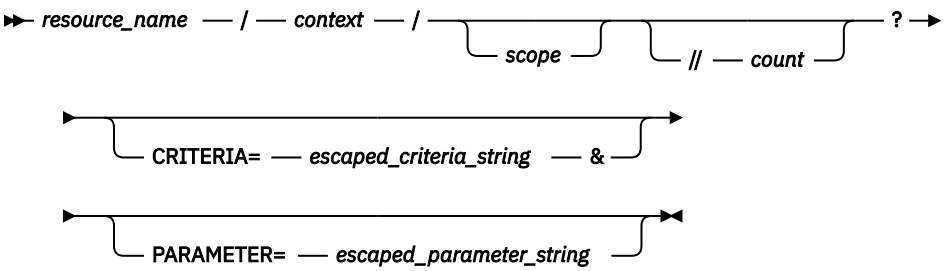
The URI of a GET request identifies the instances of the resources to be retrieved by the operation.

The following diagram illustrates the URI syntax for a GET request.

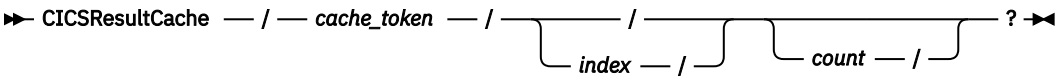
URI syntax



Resource parameters:



Cache parameters:



Options

/CICSSystemManagement
Indicates that the request is accessing the CICS management client interface. CICSSystemManagement is case-sensitive.

resource_name
Specifies the external resource name that is associated with the CICS or CICSPlex SM resource that is being queried. For example, specifying the resource name CICSRegion associates the request with the CICSPlex SM CICSRGN resource, and CICSResourceAssignmentDefinition associates the request with the RASGNDEF resource. See "CMCI resource names" on page 17 for a complete list of external resource names in CICS management client interface requests.

The HTTP GET method is not valid for all resources. To determine whether this HTTP method is valid for a particular resource, see the description of the resource in CICSPlex SM resource tables.

Resource names are not case-sensitive. If the client specifies an incorrect resource name, an HTTP 404 (Not Found) response is returned.

context
If the CMCI is installed in a CICSPlex SM environment, *context* is the name of the CICSplex or CMAS associated with the request; for example, PLEX1. See the relevant resource table in CICSPlex SM resource tables to determine whether to specify a CICSplex or CMAS.

If the CMCI is installed as a single server, *context* is the application ID of the CICS region associated with the request.

The value of context must not contain spaces. Context is not case-sensitive.

scope
Specifies the name of a CICSplex, CICS group, CICS region, or logical scope associated with the query. Scope is a subset of context, and limits the request to particular CICS systems or resources. Scope is not mandatory. If scope is absent, the request is limited by the value of the context alone. The value of scope must not contain spaces. Scope is not case-sensitive.

CICSResultCache/*cache_token*
Identifies a results cache containing a retained set of results. Specifying a cache token restricts the request to records retained from a previous GET request. At the end of every CMCI request, the CMCI checks the status of retained result sets. Any result set that has been unused for 15 minutes or more is discarded together with its *cache_token*.

index
Identifies a record in the retained results specified by the cache token parameter. Records are identified by the position they hold in the set of retained results. **index** must be an integer in the range 1 - 2 147 483 647 inclusive. A value of zero is not permitted. If you do not specify **index**, you must insert a forward slash in this position.

Specifying the **index** option does not alter the number of records requested and does not affect the warning count mechanism.

count
Identifies a subset of records in a results cache starting from the first record in the results cache or from the record specified by the **index** parameter. A negative number indicates a count back from the last record; for example, -1 means the last record, -2 the last record but one, and so on. **count** must be an integer in the range -2 147 483 647 through 2 147 483 647 inclusive. A value of zero is not permitted. In the case of resource parameters, you must precede **count** with two forward slashes.

Specifying the **count** option can limit the number of records displayed for a request, it does not alter the number of records requested and does not affect the warning count mechanism.

Query parameters
Refine the scope and nature of the request. The constituent parts of the query section can occur in any order, but each can occur only once in a URI. Query parameters are separated with an ampersand (&). Although query parameter values are not case-sensitive, certain attribute values must have the correct capitalization because some attributes such as TRANID and DESC can hold mixed-case values. The query parameters are as follows:

CRITERIA=*escaped_criteria_string*
A string of logical expressions that filters the data returned on the request. The string that makes up the value of the CRITERIA parameter follows the same rules as the filter expressions in the CICSPlex SM application programming interface. For more guidance about specifying filter expressions using the CICSPlex SM API, see How to build a filter expression.

You must replace certain restricted characters with escape sequences to ensure that they can be interpreted correctly by the server. See “CMCI escape sequences” on page 41 for guidance about using escape sequences in CICS management client interface URIs.

PARAMETER=*escaped_parameter_string*
A string of one or more parameters and values of the form **parameter_name**(*data_value*) that refines the request. The rules for specifying these parameters are the same as in the CICSPlex SM application programming interface. For more guidance about specifying parameter expressions using the CICSPlex SM API, see Performing an action against a resource.

You must encode certain characters with escape characters. See “CMCI escape sequences” on page 41 for guidance about using escape sequences in CICS management client interface URIs.

NODISCARD
Indicates that the result set is retained following the request. This attribute is specified without a value. If you do not specify NODISCARD, the result set is discarded. You can use NODISCARD for requests operating on a resource name or a retained cache. At the end of each request CMCI checks the status of retained result sets and discards any that have been unused for 15 minutes or more.

ORDERBY
Indicates that the result set is sorted by the specified fields. This attribute is specified by a string of logical expressions; the string that makes up the value of the **ORDERBY** parameter follows the same rules as the in the CICSPlex SM application programming interface. For EYUDA and CVDA values, the field is sorted on the numeric value rather than the character value displayed. For more guidance about sorting records using the CICSPlex SM API, see Sorting the records in a result set.

You must encode certain characters with escape characters. See "CMCI escape sequences" on page 41 for guidance about using escape sequences in CICS management client interface URIs.

Each request that uses this option requires an L8 TCB from the pool of L8 and L9 mode open TCBs, which is managed automatically by CICS.

OVERRIDEWARNINGCOUNT
Bypasses the warning count limit mechanism and allows the request to execute as if the default warning count limits were not specified. This option is effective only when the CICSPlex SM Web User Interface server initialization parameter **RESOURCELIMIT** is set to WARNING.

SUMMONLY
Indicates that only a result summary is returned by the request. No detailed records are included in the response.

Although specifying the **SUMMONLY** option prevents records being displayed for a request, it does not alter the number of records requested and does not affect the warning count mechanism. A request that includes **SUMMONLY** fails if the number of records requested exceeds the warning count value.

Using retained result sets to improve the performance of GET requests

You can improve the performance of HTTP GET requests by using the NODISCARD option to instruct the server to retain a set of results in memory after an initial request has completed. Instead of querying the CMAS, subsequent requests can then operate on the retained result set, which reduces the load on the server.

About this task

Normally CMCI garbage collection discards the result set after each GET request. If you use the NODISCARD option, however, the result set is retained. At the end of each request, CMCI checks the status of all result sets and discards only those that have been unused for 15 minutes or more.

The use of retained result sets permits the client to request the results of a query in several small XML response documents instead of one large one. Operating on smaller XML documents reduces the load on the server, network, and client by avoiding the need to render, transmit, and parse a large number of results. For example, by operating on retained results a client can display queried resources in pages, with each page displaying only twenty or so results from an original result set containing thousands of records. However, the snapshot nature of retained result sets makes them unsuitable for querying data that is changing rapidly or when the client needs to obtain the most up-to-date status possible.

If security is enabled on the server, you have access only to those retained result sets that you have created. An attempt to access a retained result set created by a different user results in an HTTP 403 error.

To work with a retained result set:

Procedure

1. Issue an initial HTTP GET request specifying the optional NODISCARD and SUMMONLY options in the request URI.

   Assuming that the request is successful, the CICS management client interface stores the results of the query in the GCDSA dynamic storage area and returns a token to the client that can be used on subsequent requests to access the retained result set. The token is returned as the value of

the **cachetoken** attribute in the <resultsummary> element of the GET request response. Specifying SUMMONLY prevents the server from returning any records in the body of the response.

2. Issue a second GET request specifying `CICSResultCache` in place of the external resource name, followed by the value of the **cachetoken** attribute from the response to the initial request.

   If you specify a cache token that is not recognized or has been deleted, it is treated as a resource not found condition and the server returns an HTTP 404 error.

   You can further refine the request by specifying **index**, **count,** and **orderby** options. **index** identifies one record from the result set by its position in the set of retained results. **count** identifies a subset of the records in the retained results, either from the first record in the set or from the position of the record identified by **index**.**orderby** identifies which field, or fields, the result set is sorted by. The maximum number of fields that can be specified in the **orderby** option is 32.

   You can continue to make the result set available following this request, by specifying the NODISCARD option on subsequent requests. If you do not specify NODISCARD, the server deletes the result set at the end of the request.

   The client can also issue multiple concurrent requests to the server against a single cache token. For example, multithreaded client applications can get information for records 1 - 10, 11 -20, and 21 - 30 simultaneously. However, failing to specify the NODISCARD option on one or more of the concurrent requests leads to results that are unpredictable.

Example

A GET request to view all the installed programs in a system can result in thousands of records being returned to the client, but you can use retained result sets to permit faster paging through the results.

The following request identifies all the PROGRAM resources installed in the system, but the SUMMONLY option indicates that no records are returned, only a summary of the identified resources. The NODISCARD option returns a cache token to identify this result set.

```
http://hostname.example.com:12345/CICSSystemManagement/CICSProgram/CICSPLEX?NODISCARD&SUMMONLY
```

The server returns the following response:

```
<?xml version="1.0" encoding="UTF-8"?>
<response xmlns="http://www.ibm.com/xmlns/prod/CICS/smw2int" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
connect_version="0410" version="1.0" xsi:schemaLocation="http://www.ibm.com/xmlns/prod/CICS/smw2int
http://hostname.mycompany.com:12345/CICSSystemManagement/schema/CICSSystemManagement.xsd">
<resultsummary api_response1="1024" api_response1_alt="OK" api_response2="0"
api_response2_alt="" cachetoken="C3D526A2F224FFE1" recordcount="1662"/>
</response>
```

You can verify that the query is successful by ensuring that the `api_response1` and `api_response2` attributes are 1024 and 0 respectively.

The <resultsummary> element identifies the result set with the `cachetoken` value C3D526A2F224FFE1. The `recordcount` attribute indicates that a total of 1662 records are in the result set.

To examine the first ten records in this retained result set, you can issue the following request:

```
 http://hostname.example.com:12345/CICSSystemManagement/CICSResultCache/C3D526A2F224FFE1/1/10?NODISCARD
```

The server returns the following response:

```
<?xml version="1.0" encoding="UTF-8"?>
<response xmlns="http://www.ibm.com/xmlns/prod/CICS/smw2int" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
connect_version="0410" version="1.0" xsi:schemaLocation="http://www.ibm.com/xmlns/prod/CICS/smw2int
http://hostname.mycompany.com:12345/CICSSystemManagement/schema/CICSSystemManagement.xsd">
<resultsummary api_response1="1024" api_response1_alt="OK" api_response2="0" api_response2_alt=""
cachetoken="C3D526A2F224FFE1" displayed_recordcount="10" recordcount="1662"/>
<records>
  <cicsprogram _keydata="C3C5C5C3C2D3C4E8" aloadtime="00:00:00.000" ... usecount="0" usefetch="0.000" />
  <cicsprogram _keydata="C3C5C5C3C3C9C3E2" aloadtime="00:00:00.000" ... usecount="0" usefetch="0.000" />
  <cicsprogram _keydata="C3C5C5C3D4C94040" aloadtime="00:00:00.000" ... usecount="0" usefetch="0.000" />
```

```
        ...
</records>
</response>
```

The `displayed_recordcount` attribute indicates that the <records> element contains ten <cicsprogram> child elements. The `cachetoken` attribute confirms that the result set has been retained in the server for future use.

The client can then retrieve the next ten records starting at record 11, by issuing the following request:

```
http://hostname.example.com:12345/CICSSystemManagement/CICSResultCache/C3D526A2F224FFE1/11/10?NODISCARD
```

When you have finished examining the retained result set, you inform the server by accessing the result set without specifying the NODISCARD option. The following request again uses the SUMMONLY parameter to prevent the server from returning any records on this request.

```
http://hostname.example.com:12345/CICSSystemManagement/CICSResultCache/C3D526A2F224FFE1?SUMMONLY
```

The server returns the following response:

```
<?xml version="1.0" encoding="UTF-8"?>
<response xmlns="http://www.ibm.com/xmlns/prod/CICS/smw2int" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
connect_version="0410" version="1.0" xsi:schemaLocation="http://www.ibm.com/xmlns/prod/CICS/smw2int
http://hostname.mycompany.com:12345/CICSSystemManagement/schema/CICSSystemManagement.xsd">
<resultsummary api_response1="1024" api_response1_alt="OK" api_response2="0"
api_response2_alt="" recordcount="1662"/>
</response>
```

The absence of the `cachetoken` attribute in the <resultsummary> element indicates that the result set is no longer stored in the server. If you try to issue further requests to use the retained result set, the server returns an HTTP 404 Not Found error.

CMCI POST requests

The CICS management client interface (CMCI) uses the HTTP POST method to create resources in CICS or CICSPlex SM.

The client forms a POST request from the following parts:

- The HTTP method, in this case POST
- The URI that identifies the resources to be retrieved and indicates whether the results are cached for later use or discarded
- The HTTP version
- Authorization credentials if required
- The XML body containing details of the resource to be created.

For example,

```
POST /CICSSystemManagement/CICSLocalFile/PLEX1/ HTTP/1.1
Host: example.com:22958
Authorization: Basic RlJFRDpQQVNTVzBSRA==

<request>
<create>
<parameter name="RESGROUP" value="BASIC" />
<attributes name="FILE1" defver="1" />
</create>
</request>
```

The initial line has three parts, which are separated by spaces:

- The HTTP method name
- The request URI, which is specified as an absolute path that begins `/CICSSystemManagement/` immediately followed by the external name that identifies the resource, and the parameters and filters that identify the instance of the resource that is to be created.

- The HTTP version, which, when connecting to the CICS system management client API, is always HTTP/1.1

The second line identifies the host name and port number of the target system, separated by a colon as specified in the URI. This line ends with a carriage return, followed by a line-feed.

The optional third line contains authorization credentials. If your system is running with the CICS system initialization parameter **SEC** as YES, you must supply a user ID and password in base-64 in a basic authentication header. In this example, the user ID is *FRED* and the password is *PASSWORD*.

The HTTP header is then finalized by a final carriage return and line-feed on a separate line.

The remainder of the request is the XML body specifying the attributes of the new resource. The body consists of the XML <request> element containing one <attributes> child element.

Response to a POST request

On the completion of a POST request, the client receives a response from the client API made up of an HTTP header, an XML feed containing a result summary, and details of the resources created. However, if the request included the SUMMONLY parameter, the response consists only of the HTTP header and the result summary.

The header consists of an HTTP response, the date and time, and details of the server and content type. The following example shows the header for a successful request:

```
HTTP/1.1 200 OK
Cache-Control: no-store
Date: Wed, 06 Aug 2008 08:32:00 GMT
Server: IBM_CICS_Transaction_Server (zOS)
Content-Type: application/xml; charset=UTF-8
Transfer-Encoding: chunked
```

The body of the response consists of a <response> root element containing a <resultsummary> element displaying summarized information about the request, and typically a <records> element containing details of the new resource. However, you can suppress the <records> element of the response by specifying the SUMMONLY parameter in the URI. If the request completed with errors, diagnostic information is provided in an <errors> element.

Security considerations

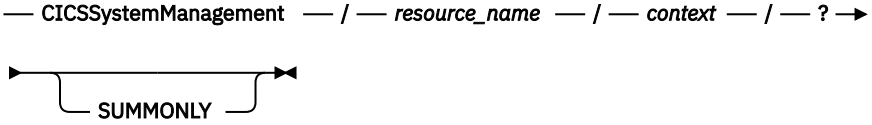
When you are using HTTP methods, be aware that some firewalls do not allow HTTP PUT or DELETE traffic through the firewall because of security considerations. To accommodate this restriction, you can use the `X-HTTP-Method-Override` HTTP header field to tunnel a PUT or DELETE request through a POST request. For example:

```
X-HTTP-Method-Override:    PUT
```

CMCI POST request URI

The URI of a POST request identifies the resources to be created by the operation.

The following diagram illustrates the URI syntax for a POST request.



In a POST request, you do not specify criteria, or parameter strings. You include all the information about the resource to be created in the body of the request.

Options

/CICSSystemManagement
Indicates that the request is accessing the CICS management client interface. CICSSystemManagement is case-sensitive.

resource_name
Specifies the external resource name that is associated with the CICS or CICSPlex SM resource that is being created. For example, specifying the resource name CICSRegion associates the request with the CICSPlex SM CICSRGN resource table, and CICSResourceAssignmentDefinition associates the request with the RASGNDEF resource table. See "CMCI resource names" on page 17 for a complete list of external resource names.

The HTTP POST method is not valid for all resources. To determine whether this HTTP method is valid for a particular resource, see the description of the resource in CICSPlex SM resource tables.

Resource names are not case-sensitive. If the client specifies an incorrect resource name, an HTTP 404 (Not Found) response is returned.

context
Specifies the name of the CICSplex or CMAS associated with the request; for example, PLEX1. This parameter is required for URIs that interact with CICSPlex SM. The context must not contain spaces. It is not case-sensitive.

SUMMONLY
Indicates that only a result summary is returned by the request. No detailed records are included in the response.

CMCI PUT requests

The CICS management client interface (CMCI) uses the HTTP PUT method to change the attributes of CICS or CICSPlex SM resources or to perform actions, including INSTALL, on those resources.

The client forms a PUT request from the following components:

- The HTTP method, in this case PUT
- The URI that identifies the resources to be changed
- The HTTP version
- Authorization credentials if required.
- The HTTP body that specifies in XML the changes to be made, or the action to be performed

For example, to update attributes of transaction definitions in CICSplex PLEX1 that have a name beginning with TR and a program beginning with P, the client constructs the following request:

```
PUT /CICSSystemManagement/CICSDefinitionTransaction/PLEX1?CRITERIA=NAME%3DTR%2A%20
AND%20PROGRAM%3DP%2A
HTTP/1.1
Host: example.com:23792
Authorization: Basic RlJFRDpQQVNTVzBSRA==

<request>
   <update>
      <attributes STATUS="ENABLED" />
   </update>
</request>
```

The initial line has three parts, separated by spaces:

- The method name
- The request URI, which is specified as an absolute path that begins `/CICSSystemManagement/` immediately followed by the external name that identifies resource together with the parameters and filters that determine which instances of that resource type are to be selected. To make the URI suitable for processing by the CICS management client interface, the client replaces certain restricted characters such as spaces and asterisks with escape sequences.

- The HTTP version, which, when connecting to the CICS management client interface, is always HTTP/1.1

The second line identifies the host name and port number of the target system separated by a colon as specified in the URI. This line ends with a carriage return, followed by a line-feed.

The optional third line contains authorization credentials. If your system is running with the CICS system initialization parameter **SEC** as YES, you must supply a user ID and password in base-64 in a basic authentication header. In this example the user ID is *FRED* and the password is *PASSWORD*.

The HTTP header is then finalized by a final carriage return and line-feed on a separate line.

The remainder of the request is the XML body specifying the change to be made to the identified resource instances. In this case the request changes the STATUS attribute of the selected transaction definitions to ENABLED. The body consists of a the XML <request> element containing either one <attributes> element or one <action> element.

A PUT request can operate on either installed resources or on definitional CICS or CICSplex SM resources. A single PUT request can either specify attribute changes or perform an action. You cannot combine attribute changes and an action in a single request.

The PUT method does not support all actions. The following actions are supported by the POST method:

- CREATE for definitional resources.
- ADD for SYSDUMP and TRANDUMP resource types.

The following actions are supported by the DELETE method:

- DISCARD for operational resources.
- REMOVE for definition resources.

Response to a PUT request

On completion of a PUT request, the client receives a response from the CICS management client interface made up of an HTTP header, and an XML feed containing a result summary, and if the request is successful, details of the changed resources.

The header consists of an HTTP response, the date and time, and details of the server and content type; for example:

```
HTTP/1.1 200 OK
Cache-Control: no-store
Date: Wed, 10 Aug 2008 12:56:00 GMT
Server: IBM_CICS_Transaction_Server (zOS)
Content-Type: application/xml; charset=UTF-8
Transfer-Encoding: chunked
```

If the PUT request fails, the server issues an HTTP non-OK response. For example if the body of the request is not valid, the header includes the following 400 response:

```
400 Bad request. The body of the request is invalid.
```

This response can occur if the client sends multiple tags on a PUT request, or the body of the request contains an unknown tag, or the tag name does not match the model name in the URI.

The body of the response consists of a <response> root element containing a <resultsummary> element displaying summarized information about the request, and typically a <records> element containing details of the selected resources. However, you can suppress the <records> element of the response by specifying the **SUMMONLY** parameter in the URI. If the request completed with errors, diagnostic information is provided in an <errors> element.

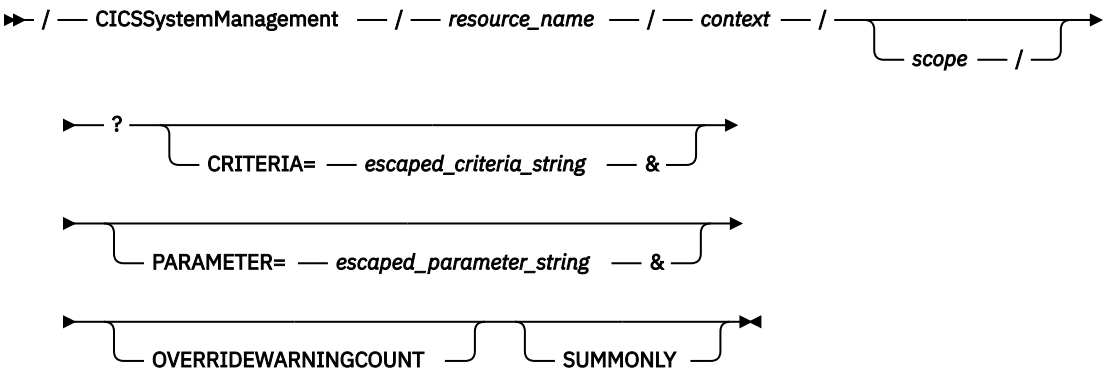
Security considerations

When you are using HTTP methods, be aware that some firewalls do not allow HTTP PUT or DELETE traffic through the firewall because of security considerations. To accommodate this restriction, you can use a CMCI POST request to tunnel a PUT request through a POST request.

CMCI PUT request URI

The URI of a PUT request identifies the instances of the resources to be targeted by the operation.

The following diagram illustrates the syntax for a PUT request URI.



Options

/CICSSystemManagement
Indicates that the request is accessing the CICS management client interface. CICSSystemManagement is case-sensitive.

resource_name
Specifies the external resource name that is associated with the CICS or CICSPlex SM resource that is being updated. For example, specifying the resource name CICSRegion associates the request with the CICSPlex SM CICSRGN resource table, and CICSResourceAssignmentDefinition associates the request with the RASGNDEF resource table. See "CMCI resource names" on page 17 for a complete list of external resource names.

The HTTP PUT method is not valid for all resources. To determine whether or not this HTTP method is valid for a particular resource, see the description of the resource in CICSPlex SM resource tables.

Resource names are not case-sensitive. If the client specifies an incorrect resource name, an HTTP 404 (Not Found) response is returned.

context
If the CMCI is installed in a CICSPlex SM environment, *context* is the name of the CICSplex or CMAS associated with the request; for example, PLEX1.

If the CMCI is installed as a single server, *context* is the application ID of the CICS region associated with the request.

The value of *context* must not contain spaces. It is not case-sensitive.

scope
Specifies the name of a CICSplex, CICS group, CICS region, or logical scope associated with the request. The scope is a subset of the context and limits the request to particular CICS systems or resources. Scope is not mandatory. If it is absent, the request is limited by the value of the context alone. The scope must not contain spaces. It is not case-sensitive.

Query parameters
Refine the scope and nature of the request. The constituent parts of the query section can occur in any order, but each can occur only once in a URI. Query parameters are separated with an ampersand (&). Although query parameter values are not case-sensitive, certain attribute values must have

the correct capitalization because some attributes such as TRANID and DESC can hold mixed-case values. The query parameters are as follows:

CRITERIA=*escaped_criteria_string*
A string of logical expressions that filters the data returned on the request. The string that makes up the value of the CRITERIA parameter follows the same rules as the filter expressions in the CICSPlex SM application programming interface. For more guidance about specifying filter expressions using the CICSPlex SM API, see How to build a filter expression.

You must replace certain restricted characters with escape sequences to ensure that they can be interpreted correctly by the server. See "CMCI escape sequences" on page 41 for guidance about using escape sequences in CICS management client interface URIs.

PARAMETER=*escaped_parameter_string*
A string of one or more parameters and values of the form **parameter_name**(*data_value*) that refines the request. The rules for specifying these parameters are the same as in the CICSPlex SM application programming interface. For more guidance about specifying parameter expressions using the CICSPlex SM API, see Performing an action against a resource.

As with the criteria string, you must encode certain characters with escape characters. See "CMCI escape sequences" on page 41 for guidance about using escape sequences in CICS management client interface URIs.

OVERRIDEWARNINGCOUNT
Bypasses the warning count limit mechanism and allows the request to execute as if the default warning count limits were not specified. This option is effective only when the CICSPlex SM Web User Interface server initialization parameter **RESOURCELIMIT** is set to WARNING.

SUMMONLY
Indicates that only a result summary is returned by the request. No detailed records are included in the response.

Although specifying the **SUMMONLY** option prevents records being displayed for a request, it does not alter the number of records requested and does not affect the warning count mechanism. A request that includes **SUMMONLY** fails if the number of records requested exceeds the warning count value.

CMCI resource names

The CICS management client interface (CMCI) uses external resource names that map to CICSPlex SM and CICS resources.

The following tables list the CMCI external resource names and their associated CICSPlex SM resource names. The resources are divided into CICS operational resources, CICS definitional resources, CICSPlex SM definitional resources, and CICSPlex SM nondefinitional resources.

See CICSPlex SM resource tables for information about CICSPlex SM resources including the CMCI methods that are valid for each resource.

CICS operational resources

Table 1. Resource names of CICS operational resources

External resource name	CICSPlex SM resource name	Description
CICSAllTaskSubpools	TSKSPOLS	CICS task storage subpools
CICSAtomService	ATOMSERV	CICS Atom service, feed, collection, or category document
CICSAutoInstallModel	AIMODEL	CICS autoinstall terminal model
CICSBridgeFacility	BRFACIL	Virtual terminal (bridge facility) used by the 3270 bridge mechanism
CICSBundle	BUNDLE	CICS bundle
CICSBundlePart	BUNDPART	CICS bundle part
CICSCaptureSpecification	EVCSPEC	CICS event capture specifications
CICSCaptureSpecificationDataPredicate	EVCSDATA	CICS event capture specification data predicates
CICSCaptureSpecificationInformationSource	EVCSINFO	CICS event capture specification information sources
CICSCaptureSpecificationOptionPredicate	EVCSOPT	CICS event capture specification command option predicates
CICSCFDTPool	CFDTPOOL	Coupling facility data table pool
CICSDataSetName	DSNAME	Data set in use by an active CICS system
CICSDataTable	CMDT	Files that have CICS or user-maintained data tables, or coupling facility data tables, associated with them
CICSDb2Connection	DB2CONN	Db2® connection
CICSDb2Entry	DB2ENTRY	Db2 entry
CICSDb2Transaction	DB2TRN	Db2 transaction
CICSDBCTLSubsystem	DBCTLSS	Connections of an active CICS system to a DBCTL subsystem

Table 1. Resource names of CICS operational resources (continued)		
External resource name	**CICSPlex SM resource name**	**Description**
CICSDFHRPLDataSet	RPLLIST	Data set in the CICS DFHRPL concatenation sequence
CICSDocumentTemplate	DOCTEMP	Document template
CICSDomainSubpool	DOMSPOOL	CICS storage domain subpool
CICSDynamicStorageArea	CICSDSA	Dynamic storage area
CICSEnqueue	ENQUEUE	CICS enqueue
CICSEnqueueModel	ENQMODEL	ENQ/DEQ model
CICSEPAdapter	EPADAPT	CICS event processing adapter
CICSEPAdapterSet	EPADSET	CICS event processing adapter set
CICSEPAdapterInSet	EPAINSET	CICS event processing adapters in an event processing adapter set
CICSEventBinding	EVNTBIND	CICS event bindings
CICSEventProcessing	EVNTGBL	CICS event processing
CICSEXCIRequest	EXCI	Task that originated from client programs using the external CICS interface API
CICSExtrapartitionTDQueue	EXTRATDQ	Extrapartition transient data queue
CICSFEPIConnection	FEPICONN	FEPI connection
CICSFEPINode	FEPINODE	FEPI node
CICSFEPIPool	FEPIPOOL	FEPI pool
CICSFEPIPropertySet	FEPIPROP	FEPI property set
CICSFEPITarget	FEPITRGT	FEPI target
CICSGlobalDispatcher	DSPGBL	Global CICS dispatcher information for CICS systems.
CICSGlobalDynamicStorageArea	CICSSTOR	CICS dynamic storage areas
CICSGlobalMVSTCBStatistics	MVSTCBGL	Global MVS™ TCBs in the CICS address space
CICSGlobalTDQueueStatistics	TDQGBL	Intrapartition transient data queue usage
CICSGlobalTSQueueStatistics	TSQGBL	Temporary storage queue usage
CICSGlobalURIMapStatistics	URIMPGBL	Global statistics returned by CICS extract statistics for the URIMAP resource
CICSGlobalUserExit	EXITGLUE	Installed CICS TS global user exits
CICSIndirectTDQueue	INDTDQ	Indirect transient data queue
CICSIntervalControlRequest	REQID	Outstanding interval control requests
CICSIntrapartitionTDQueue	INTRATDQ	Intrapartition transient data queue

Table 1. Resource names of CICS operational resources (continued)

External resource name	CICSPlex SM resource name	Description
CICSIPFacility	IPFACIL	Associations between active CICS tasks and the IPIC connections in use by those tasks
CICSIPICConnection	IPCONN	IP intercommunications connection
CICSISCMROConnection	CONNECT	ISC or MRO connection
CICSJournalModel	JRNLMODL	Installed journal model and corresponding log stream name
CICSJournalName	JRNLNAME	Information about the system log and general logs
CICSJVM	JVM	Java™ virtual machine
CICSJVMClassCache	CLCACHE	Shared class cache
CICSJVMPool	JVMPOOL	JVM pool
CICSJVMProfile	JVMPROF	JVM profiles
CICSJVMServer	JVMSERV	Installed JVM servers
CICSLIBRARY	LIBRARY	LIBRARY
CICSLIBRARYDataSetName	LIBDSN	LIBRARY data set names
CICSLoader	LOADER	CICS loader information
CICSLoaderByDSA	LOADACT	CICS loader activity by dynamic storage area
CICSLocalFile	LOCFILE	CICS local file
CICSLocalTransaction	LOCTRAN	CICS local transaction
CICSLSRPool	LSRPOOL	Local shared resources pool
CICSLSRPoolBuffer	LSRPBUF	VSAM LSR pool buffer
CICSLU62ModeName	MODENAME	LU 6.2 mode name
CICSMonitoringAndStatistics	MONITOR	CICS monitoring and statistics information
CICSMQConnection	MQCON	IBM MQ connection
CICSMQConnectionStatistics	MQCONN	IBM MQ connection statistics
CICSMQInitiationQueue	MQINI	IBM MQ initiation queue
CICSMVSLogStream	STREAMNM	MVS log stream name
CICSMVSStorageArea	MVSESTG	MVS storage area
CICSMVSTCB	MVSTCB	MVS TCBs
CICSMVSWLM	MVSWLM	MVS workload management component information
CICSOSGIBundle	OSGIBUND	OSGi bundles
CICSOSGIService	OSGISERV	OSGi services

Table 1. Resource names of CICS operational resources (continued)		
External resource name	**CICSPlex SM resource name**	**Description**
CICSPagePool	CICSPAGP	CICS page (DSA) pool
CICSPartner	PARTNER	CPI-C partner table
CICSPipeline	PIPELINE	CICS web service processing nodes
CICSProcessType	PROCTYP	BTS process type
CICSProfile	PROFILE	CICS profile
CICSProgram	PROGRAM	CICS program
CICSRecoveryManager	RECOVERY	Recovery manager information
CICSRegion	CICSRGN	CICS region
CICSRemoteFile	REMFILE	Remote CICS file
CICSRemoteTDQueue	REMTDQ	Remote transient data queue
CICSRemoteTransaction	REMTRAN	Remote transaction
CICSSharedTSQueue	TSQSHR	Temporary storage queue
CICSSystemDump	SYSDUMP	CICS system dump code
CICSSystemParameter	SYSPARM	CICS system parameter
CICSTask	TASK	CICS task
CICSTaskAssociation	TASKASSC	Task association data
CICSTaskFileUsage	TASKFILE	Tasks and the CICS files they have used
CICSTaskRelatedExit	EXITTRUE	Task-related user exit
CICSTaskRMIUsage	TASKRMI	CICS resource manager interface usage by task
CICSTaskStorage	TASKESTG	CICS storage element for a task
CICSTaskSubpool	TSKSPOOL	CICS task storage subpool
CICSTaskTSQueueUsage	TASKTSQ	Tasks and the CICS temporary storage queues they have used
CICSTCBMode	DSPMODE	CICS dispatcher TCB mode
CICSTCBPool	DSPPOOL	CICS dispatcher TCB pool
CICSTCPIPService	TCPIPS	TCP/IP service
CICSTCPIPStatistics	TCPIPGBL	CICS internal TCP/IP sockets support
CICSTerminal	TERMNL	CICS terminal
CICSTransactionClass	TRANCLAS	CICS transaction class
CICSTransactionDump	TRANDUMP	CICS transaction dump code
CICSTSModel	TSMODEL	CICS temporary storage model
CICSTSPool	TSPOOL	CICS temporary storage pool

Table 1. Resource names of CICS operational resources (continued)		
External resource name	**CICSPlex SM resource name**	**Description**
CICSTSQueue	TSQNAME	CICS nonshared temporary storage queue
CICSUOW	UOW	CICS unit of work
CICSUOWEnqueue	UOWENQ	Enqueue held by a unit of work
CICSUOWLink	UOWLINK	Link between a unit of work and a CICS system
CICSUOWShuntedAndHoldingLocks	UOWDSNF	Shunted unit of work
CICSURIHost	HOST	Virtual hosts in the local system
CICSURIMap	URIMAP	Universal resource identifier request
CICSWebService	WEBSERV	Information about the runtime environment for a CICS application program deployed in a web services setting
CICSWorkRequest	WORKREQ	EJB work request
CICSXmltransform	XMLTRANS	XML transforms

CICS definitional resources

Table 2. Resource names of CICS definitional resources		
External resource name	**CICSPlex SM resource name**	**Description**
CICSCSDGroup	CSDGROUP	CSD group definition
CICSCSDGroupInList	CSDINLST	CSD groups in list
CICSCSDList	CSDLIST	CSD list definition
CICSCSDResource	CSDINGRP	CSD CICS resources in group
CICSDefinitionAtomService	ATOMDEF	Atom service definition
CICSDefinitionBundle	BUNDDEF	Bundle definition
CICSDefinitionCorbaServer	EJCODEF	CorbaServer definition
CICSDefinitionDb2Connection	DB2CDEF	Db2 connection definition
CICSDefinitionDb2Entry	DB2EDEF	Db2 entry definition
CICSDefinitionDb2Transaction	DB2TDEF	Db2 transaction definition
CICSDefinitionDeployedJARFile	EJDJDEF	CICS-deployed JAR file definition
CICSDefinitionDocumentTemplate	DOCDEF	Document template definition
CICSDefinitionEnqueueModel	ENQMDEF	Global enqueue definition
CICSDefinitionFEPINode	FENODDEF	FEPI node definition
CICSDefinitionFEPIPool	FEPOODEF	FEPI pool definition
CICSDefinitionFEPIPropertySet	FEPRODEF	FEPI property set definition

Table 2. Resource names of CICS definitional resources (continued)		
External resource name	**CICSPlex SM resource name**	**Description**
CICSDefinitionFEPITarget	FETRGDEF	FEPI target definition
CICSDefinitionFile	FILEDEF	CICS file definition
CICSDefinitionIPICConnection	IPCONDEF	IPIC connection definition
CICSDefinitionISCMROConnection	CONNDEF	ISC/MRO connection definition
CICSDefinitionJournalModel	JRNMDEF	Journal model definition
CICSDefinitionLIBRARY	LIBDEF	LIBRARY definition
CICSDefinitionLSRPool	LSRPOOL	Local shared resources pool definition
CICSDefinitionMapSet	MAPDEF	Map set definition
CICSDefinitionMQConnection	MQCONDEF	IBM MQ connection definition
CICSDefinitionPartitionSet	PRTNDEF	Partition set definition
CICSDefinitionPartner	PARTDEF	Partner definition
CICSDefinitionPipeline	PIPEDEF	Pipeline definition
CICSDefinitionProcessType	PROCDEF	Process type definition
CICSDefinitionProfile	PROFDEF	Profile definition
CICSDefinitionProgram	PROGDEF	Program definition
CICSDefinitionRequestModel	RQMDEF	Request model definition
CICSDefinitionSession	SESSDEF	Session definition
CICSDefinitionTCPIPService	TCPDEF	TCP/IP service definition
CICSDefinitionTDQueue	TDQDEF	Transient data queue definition
CICSDefinitionTerminal	TERMDEF	Terminal definition
CICSDefinitionTransaction	TRANDEF	Transaction definition
CICSDefinitionTransactionClass	TRNCLDEF	Transaction class definition
CICSDefinitionTSModel	TSMDEF	Temporary storage model definition
CICSDefinitionTypeterm	TYPTMDEF	Typeterm definition
CICSDefinitionURIMap	URIMPDEF	URI mapping definition
CICSDefinitionWebService	WEBSVDEF	Web service definition

CICSPlex SM definitional resources

Table 3. Resource names of CICSPlex SM definitional resources		
External resource name	**CICSPlex SM resource name**	**Description**
CICSActiveAnalysisPointSpecification	APSPEC	RTA analysis point specification
CICSApplicationDefinition	APPLDEF	Application definition
CICSBatchrepRequest	BATCHREP	Batched repository-update job

Table 3. Resource names of CICSPlex SM definitional resources (continued)

External resource name	CICSPlex SM resource name	Description
CICSCICSplexDefinition	CPLEXDEF	CICSplex definition
CICSCMASToCMASLinkDefinition	CMTCMDEF	CMAS-to-CMAS link definition
CICSMonitorDefinition	MONDEF	Monitor definitions
CICSMonitorGroup	MONGROUP	Monitor groups
CICSMonitorGroupInSpecification	MONINSPC	Monitor groups in monitor specifications
CICSMonitorResourceInGroup	MONINGRP	Monitor definitions in groups
CICSMonitorSpecification	MONSPEC	Monitor specifications
CICSMonitorSpecificationsToSystem	LNKSMSCS	Monitor specifications to CICS system links
CICSMonitorSpecificationsToSystemGroup	LNKSMSCG	Monitor specifications to system group links
CICSPeriodDefinition	PERIODEF	Time period definition
CICSPlatformDefinition	PLATDEF	Platform definition
CICSPrimaryAnalysisPointCMAS	CMDMPAPS	Primary CMAS analysis point specification
CICSRegionDefinition	CSYSDEF	CICS region definition
CICSRegionGroup	CSYSGRP	CICS system group definition
CICSResourceAssignmentDefinition	RASGNDEF	Resource assignment definition
CICSResourceAssignmentInDescription	RASINDSC	Resource assignments in resource description
CICSResourceDescription	RESDESC	Resource description definition
CICSResourceGroup	RESGROUP	Resource group definition
CICSResourceInGroup	RESINGRP	Resources associated with a resource group
CICSResourceGroupInDescription	RESINDSC	Resource groups in resource description
CICSRTAActionDefinition	ACTION	RTA action definition
CICSRTADefinition	RTADEF	RTA definition
CICSRTAEvaluationDefinition	EVALDEF	Evaluation definitions
CICSRTAGroup	RTAGROUP	RTA group definition
CICSRTAResourceInGroup	RTAINGRP	RTA definition in an RTA group
CICSRTASpecification	RTASPEC	RTA specification definition
CICSRTAStatusDefinition	STATDEF	RTA status definition
CICSRTAStatusDefinitionInGroup	STAINGRP	Status definition in an RTA group

Table 3. Resource names of CICSPlex SM definitional resources (continued)		
External resource name	**CICSPlex SM resource name**	**Description**
CICSSecondaryAnalysisPointCMAS	CMDMSAPS	Secondary CMAS analysis point specification
CICSSystemGroupToSystemGroup	CSGLCGCS	CICS system to group links
CICSSystemLink	SYSLINK	CICS system link definition
CICSSystemToSystemGroup	CSGLCGCG	CICS system group to group links
CICSTransactionGroup	TRANGRP	CICS transaction group definition
CICSTransactionInGroup	DTRINGRP	Transactions in transaction groups
CICSWLMActiveRouter	WLMAROUT	CICS router region in an active workload
CICSWLMDefinition	WLMDEF	WLM definition
CICSWLMGroup	WLMGROUP	WLM group definition
CICSWLMResourceInGroup	WLMINGRP	WLM definition in a workload group
CICSWLMSpecification	WLMSPEC	WLM specification

CICSPlex SM nondefinitional resources

Table 4. Resource names of CICSPlex SM nondefinitional resources		
External resource name	**CICSPlex SM resource name**	**Description**
CICSActiveAnalysisPoint	APACTV	Installed analysis definitions associated with an analysis point specification
CICSApplication	APPLCTN	Deployed application
CICSCICSManagingAddressSpace	CMAS	Active CMAS
CICSCICSManagingAddressSpaceList	CMASLIST	CMASs known to the local CMAS
CICSCICSplex	CICSPLEX	CICSplex being managed by a CMAS
CICSCICSplexList	CMASPLEX	All CICSplexes being managed by local CMAS
CICSCMASInCICSplex	CPLXCMAS	CMAS in CICSplex definition
CICSCMASToCMASLink	CMTCMLNK	CMAS-to-CMAS link
CICSManagedRegion	MAS	CICSPlex SM managed CICS region
CICSManagedRegionStatus	MASSTAT	Status of a managed CICS region by CMAS
CICSManagementPart	MGMTPART	Management part for an application or platform
CICSPlatform	PLATFORM	Platform
CICSResourceByAssignment	RASPROC	Resource selected by a resource assignment
CICSResourceByDescription	RDSCPROC	Resource selected by a resource description
CICSResourceBySystem	SYSRES	Resource assigned to CICS systems

Table 4. Resource names of CICSPlex SM nondefinitional resources (continued)

External resource name	CICSPlex SM resource name	Description
CICSRTAEvent	EVENT	Significant outstanding change in the status of a CICSplex or one of its CICS systems
CICSRTAEventDetail	EVENTDTL	Evaluation that participated in the generation of an event
CICSRTAInstalledDefinition	RTAACTV	RTA installed analysis and status definition
CICSRule	RULE	Policy rule information
CICSTaskHistory	HTASK	Completed task
CICSTaskHistoryCollection	MASHIST	CICS task history collection
CICSTopologyAtomService	CRESATOM	Atom service in a CICS system
CICSTopologyAutoInstallModel	CRESAIMD	Topology data for autoinstall model
CICSTopologyBundle	CRESBUND	Bundle in a CICS system
CICSTopologyBTSProcessType	CRESPRTY	BTS Process Type in a CICS system
CICSTopologyCaptureSpecification	CRESEVCS	Event Processing capture specification
CICSTopologyDataSet	CRESDSNM	Data set in a CICS system
CICSTopologyDb2Connection	CRESDB2C	Db2 connection in a CICS system
CICSTopologyDb2Entry	CRESDB2E	Db2 entry in a CICS system
CICSTopologyDb2Transaction	CRESDB2T	Db2 transaction in a CICS system
CICSTopologyDocumentTemplate	CRESDOCT	Document template in a CICS system
CICSTopologyEnqueueModel	CRESENQM	ENQ/DEQ model in a CICS system
CICSTopologyEPAdapter	CRESEPAD	Event processing adapter
CICSTopologyEPAdapterSet	CRESEPAS	Event processing adapter set
CICSTopologyEventBinding	CRESEVBD	Event binding in a CICS system
CICSTopologyFEPIConnection	CRESFECO	FEPI connection in a CICS system
CICSTopologyFEPINode	CRESFEND	FEPI node in a CICS system
CICSTopologyFEPIPool	CRESFEPO	FEPI pool in a CICS system
CICSTopologyFEPITarget	CRESFETR	FEPI target in a CICS system
CICSTopologyFile	CRESFILE	File in a CICS system
CICSTopologyGlobalUserExit	CRESGLUE	Global user exit in a CICS system
CICSTopologyIPICConnection	CRESIPCN	IPIC connection in a CICS system
CICSTopologyISCMROConnection	CRESCONN	ISC/MRO connection in a CICS system
CICSTopologyJournalName	CRESJRNM	Journal name in a CICS system
CICSTopologyJVMServer	CRESJVMS	JVM server in a CICS system
CICSTopologyLibrary	CRESLIBR	Topology data for LIBRARY
CICSTopologyLU62ModeName	CRESMODE	LU62 mode name in a CICS system

Table 4. Resource names of CICSPlex SM nondefinitional resources (continued)		
External resource name	**CICSPlex SM resource name**	**Description**
CICSTopologyPartner	CRESPART	Partner in a CICS system
CICSTopologyPipeline	CRESPIPE	Topology data for PIPELINE
CICSTopologyProfile	CRESPROF	Profile in a CICS system
CICSTopologyProgram	CRESPRGM	Program in a CICS system
CICSTopologyRequestModel	CRESRQMD	Request model in a CICS system
CICSTopologySystemDump	CRESSDMP	System dump in a CICS system
CICSTopologyTaskRelatedExit	CRESTRUE	Task-related exit in a CICS system
CICSTopologyTCPIPService	CRESTCPS	TCP/IP service in a CICS system
CICSTopologyTDQueue	CRESTDQ	Transient data queue in a CICS system
CICSTopologyTerminal	CRESTERM	Terminal in a CICS system
CICSTopologyTransaction	CRESTRAN	Transaction in a CICS system
CICSTopologyTransactionDump	CRESTDMP	Transaction dump in a CICS system
CICSTopologyTSModel	CRESTSMD	Temporary storage model in a CICS system
CICSTopologyUrimap	CRESURIM	Topology data for URIMAP
CICSTopologyWebservice	CRESWEBS	Topology data for WEBSERVICE
CICSTopologyXMLTransform	CRESXMLT	Topology data for XML transform
CICSWLMActiveAffinity	WLMATAFF	Active workload transaction group affinity
CICSWLMActiveAOR	WLMAWAOR	Active workload target distribution factors
CICSWLMActiveDefinition	WLMAWDEF	Active workload definition
CICSWLMActiveTOR	WLMAWTOR	Active workload routing region
CICSWLMActiveTransactionGroup	WLMATGRP	Active workload transaction groups
CICSWLMActiveTransaction	WLMATRAN	Active workload dynamic transaction
CICSWLMActiveWorkload	WLMAWORK	Active workload
CICSWLMTarget	WLMATARG	Target region for one or more active workloads

CMCI XML

CMCI requests and responses contain information in XML format. The unique XML elements in the CMCI are defined in a schema named CICSSystemManagement. This schema is constructed by incorporating a separate schema for each resource type.

Client applications need to access the CICSSystemManagement schema to validate and format the information in the body of responses to CICS management client interface requests. Clients can access the schema that defines this output XML at the following URI: `http://hostname:portnumber/CICSSystemManagement/schema/CICSSystemManagement.xsd`.

Note: CMCI can interpret only characters that are handled by the EBCDIC 037 code page. Using other characters in CMCI requests leads to unpredictable results.

<request> element

The <request> element is the root element for the body XML of a CICS management client interface POST or PUT request.

In a POST request, the <request> element contains one child <create> element. In a PUT request, the <request> element can contain one child <update> element or one <action> element, as shown in the example.

In a DELETE request, the <request> element is typically empty. If specified, the <request> element must contain one child <delete> element.

A CMCI request can contain only one <request> element.

Contains:

In a POST request: one “<create> element” on page 27
In a PUT request: one “<update> element” on page 28 or one “<action> element” on page 29
In a DELETE request: one “<delete> element” on page 30

Example

The following example shows an install action for the BAS resource with name ABCD. Notice that the <action> element is contained in a <request> element.

```
PUT http://exampledomain.com:12345/CICSSystemManagement/<RESOURCE NAME>/<CONTEXT>/?CRITERIA=NAME%3DABCD

   <request>
      <action name="INSTALL">
         <parameter name="TARGET" value="SYSTEM1"/>
         <parameter name="USAGE" value="LOCAL"/>
      </action>
   </request>
```

<create> element

The <create> element contains the <parameter> and <attributes> elements in a POST request.

Contained by:

“<request> element” on page 27

Contains:

The <create> element must contain one <attributes> element and can contain one or more child <parameter> elements. A POST request can contain only one <create> element.

“<parameter> element” on page 31
“<attributes> element” on page 31

Examples

The following example shows the creation of a BAS resource with name ABCD.

```
POST http://exampledomain.com:12345/CICSSystemManagement/<RESOURCE NAME>/<CONTEXT>

   <request>
      <create>
         <attributes name="ABCD" defver="1" />
      </create>
   </request>
```

The following example shows the creation of a BAS resource with name ABCD in the group GROUP1.

```
POST http://exampledomain.com:12345/CICSSystemManagement/<RESOURCE NAME>/<CONTEXT>

   <request>
      <create>
         <parameter name="RESGROUP" value="GROUP1"/>
         <attributes name="ABCD" defver="1" />
      </create>
   </request>
```

The following example shows the creation of a CSD resource with name ABCD in the group GROUP2.

```
POST http://exampledomain.com:12345/CICSSystemManagement/<RESOURCE NAME>/<CONTEXT>/<SCOPE>

   <request>
      <create>
         <parameter name="CSD"/>
         <attributes name="ABCD" csdgroup="GROUP2"/>
      </create>
   </request>
```

<update> element

The <update> element contains the <parameter> and <attributes> elements in a PUT request.

Contained by:

“<request> element” on page 27

Contains:

The <update> element must contain one <attributes> element and can contain one or more child <parameter> elements. A PUT request can contain only one <update> element.

“<parameter> element” on page 31
“<attributes> element” on page 31

Example

The following example updates the BAS Transaction definition with name ABCD to have values `shutdown="DISABLED"`, `isolate="YES"`, and `dynamic="NO"`.

```
PUT http://exampledomain.com:12345/CICSSystemManagement/CICSDefinitionalTransaction/<CONTEXT>/?
CRITERIA=NAME%3DABCD

   <request>
      <update>
         <attributes shutdown="DISABLED" isolate="YES" dynamic="NO" />
      </update>
   </request>
```

The following example updates the `description` of the CSD resource with name ABCD in group GROUP2.

```
PUT http://exampledomain.com:12345/CICSSystemManagement/<RESOURCE NAME>/<CONTEXT>/<SCOPE>
    /?CRITERIA=NAME%3DABCD&PARAMETER=CSDGROUP%28GROUP2%29

   <request>
      <update>
         <parameter name="CSD"/>
         <attributes description="updated"/>
      </update>
   </request>
```

<action> element

The <action> element contains details of an action to be performed by a CICS management client interface PUT request on the resources specified by the URI.

Contained by:

"<request> element" on page 27

Attributes

name="*action_name*"

The <action> element must contain only one **name** attribute identifying the action to be performed.

You must enclose *action_name* in matching single or double quotes. As is the convention in coding XML, you can use escape sequences to replace certain special characters such as &, <, and >. If the action has any parameters associated with it, you must specify them as child elements of <action> using the <parameter> element.

If you specify the <action> element in the HTTP body, you must not include an <attributes> element in the same request.

Contains:

The <action> element can contain one or more child <parameter> elements.

"<parameter> element" on page 31

Examples

The following example shows an install for the BAS resource with name ABCD.

```
PUT http://exampledomain.com:12345/CICSSystemManagement/<RESOURCE NAME>/<CONTEXT>/?CRITERIA=NAME%3DABCD

   <request>
      <action name="INSTALL">
         <parameter name="TARGET" value="SYSTEM1"/>
         <parameter name="USAGE" value="LOCAL"/>
      </action>
   </request>
```

The following example shows a CSDADD action where the CSD group, GROUP4, is added to the CSD list, LIST2, after the CSD group, GROUP3.

The parameters that are specified for the CSDADD action are: `TO_CSDLIST`, `ADD_CSDGROUP`, and `ADD_LOCATION`.

`TO_CSDLIST` specifies which CSD list to add the CSD group to.
`ADD_CSDGROUP` specifies which CSD group to add the supplied CSD group next to in the CSD list.
`ADD_LOCATION` specifies whether to add the supplied CSD group `"BEFORE"` or `"AFTER"` the existing CSD group in the CSD list.

```
PUT http://exampledomain.com:12345/CICSSystemManagement/CICSCSDGroup/<CONTEXT>/<SCOPE>/?
CRITERIA=NAME%3DGROUP4
```

```
   <request>
      <action name="CSDADD">
         <parameter name="TO_CSDLIST" value="LIST2"/>
         <parameter name="ADD_CSDGROUP" value="GROUP3"/>
         <parameter name="ADD_LOCATION" value="AFTER"/>
      </action>
   </request>
```

The following example shows a DISABLE action on the CICS operational resource, RESOURCE. Note that the type of resource is included in the URI, in this case the resource is of type `CICSAtomService`.

```
PUT http://exampledomain.com:12345/CICSSystemManagement/CICSAtomService/<CONTEXT>/<SCOPE>/?
CRITERIA=NAME%3DRESOURCE

   <request>
      <action name="DISABLE"/>
   </request>
```

<delete> element

The <delete> element contains details of a delete to be performed by a CICS management client interface DELETE request on the resources specified by the URI.

Contained by:

"<request> element" on page 27

Contains:

Typically a <delete> element does not contain any parameters, and therefore is not required.

"<parameter> element" on page 31

If you are deleting a CSD definitional resource, you must include PARAMETER=CSDGROUP in the URI. You can also include `<parameter name="CSD"/>` in the XML body but this is not required. If you have specified PARAMETER=CSDGROUP in the URI and you do not specify `<parameter name="CSD"/>` in the XML body, CICS assumes that `<parameter name="CSD"/>` has been specified.

Examples

The following example shows a delete request for a BAS resource definition with the name ABCD. Only the URI is required for a delete request.

```
DELETE /CICSSystemManagement/<RESOURCE_NAME>/<CONTEXT>/?CRITERIA=NAME%3DABCD
```

The following example shows a delete request for a CSD resource definition. Notice that PARAMETER=CSDGROUP is contained in the URI.

```
DELETE /CICSSystemManagement/<RESOURCE_NAME>/<CONTEXT>/<SCOPE>?
CRITERIA=NAME%3DABCD&PARAMETER=CSDGROUP%28GROUP1%29
```

The following example is equivalent to the previous one, the difference being that the optional <delete> element has been included in the XML body.

```
DELETE /CICSSystemManagement/<RESOURCE_NAME>/<CONTEXT>/<SCOPE>?
CRITERIA=NAME%3DABCD&PARAMETER=CSDGROUP%28GROUP1%29

   <request>
      <delete>
         <parameter name="CSD"/>
      </delete>
   </request>
```

<parameter> element

The <parameter> element contains the names and values of any parameters associated with a CICS management client interface POST or PUT requests. A POST or PUT request can contain one or more parameter elements.

Contained by:

In a POST request: “<create> element” on page 27
In a PUT request: one “<action> element” on page 29 or one “<update> element” on page 28

Attributes

name="*parameter_name*"
The name of a parameter associated with an action. *parameter_name* is case-sensitive and must be enclosed in matching single or double quotes.

value="*parameter_value*"
The value of the named parameter. *parameter_value* is case-sensitive and must be enclosed in matching single or double quotes.

Example

```
<parameter name="TARGET" value="SYSTEM1"/>
<parameter name="USAGE" value="LOCAL"/>
```

<attributes> element

The <attributes> element contains details of the new values of attributes in a CICS management client interface POST or PUT request. A request can contain only one <attributes> element.

Contained by:

“<request> element” on page 27

Attributes

***attribute_name*="*data_value*"**
The <attributes> element must contain one or more *attribute_name* "*data_value*" combinations specifying the new values to be given to the named attributes. See CICSPlex SM resource tables for a list of required attributes for each resource.

The *data_value* attribute is case-sensitive and must be enclosed in matching single or double quotes. As is the convention in coding XML, you can use escape sequences to replace certain special characters such as &, <, and >.

If you specify the <attributes> element in the HTTP body, you must not include an <action> element in the same request.

Example

```
<attributes name=”ABCD” defver=”1” status=”ENABLED” taskdataloc=”ANY”
taskdatakey=”USER” storageclear=”NO” shutdown=”DISABLED” isolate=”YES”
dynamic=”NO” routable=”NO” restart=”NO” spurge=”NO” tpurge=”NO” dump=”YES”
trace=”YES” otstimeout=”NO” wait=”YES” ressec=”NO” cmdsec=”NO”
program=”ABCDEFGH” />
```

<response> element

The <response> element is the root element in the response to a CICS management client interface request. The <response> element can contain up to three child elements; a mandatory <resultsummary> element, a <records> element containing any records returned by the request, and, if the request returns errors, an <errors> element containing diagnostic information.

Contains:

"<resultsummary> element" on page 32
"<errors> element" on page 34
"<records> element" on page 34

Attributes

xmlns="http://www.ibm.com/xmlns/prod/CICS/smw2int"
The CICS management client interface namespace.

xsi:schemaLocation="http://www.ibm.com/xmlns/prod/CICS/smw2int http://*hostname:portnumber*/CICSSystemManagement/schema/CICSSystemManagement.xsd"
The location of the CICS management client interface XML schema.

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
The XML schema instance namespace.

version="*schema_version*"
The version number of the CICS management client interface schema. This number starts at 1.0 and is incremented each time the schema is updated.

connect_version="*connect_version*"
The CICSPlex SM release number that was used when connecting to CICSPlex SM using the EXEC CPSM CONNECT API command.

Example

```
<response xmlns="http://www.ibm.com/xmlns/prod/CICS/smw2int" xsi:schemaLocation=
"http://www.ibm.com/xmlns/prod/CICS/smw2int http://example.com:27231/
CICSSystemManagement/schema/CICSSystemManagement.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.0"
connect_version="0410">


....
</response>
```

<resultsummary> element

The <resultsummary> element contains summarized information about the CICS management client interface request. The response to a CICS management client interface request must contain a <resultsummary> element.

Contained by:

"<response> element" on page 32

Attributes

api_source="*value*"
If the request is not successful, contains the name of the API source; for example, "CICSPlex SM" or "CMCI". This attribute is not present in the response to a successful request.

api_function="*function*"
If the request is not successful, contains the name of the CICS system command or CICSPlex SM API command that has failed; for example, PERFORM SET. This attribute is not present in the response to a successful request.

api_response1="*value*"
The CICS EIBRESP code or the CICSPlex SM API EYUDA response code as a numeric value. See RESP and RESP2 options for more information about CICS RESP values and EYUDA values for information about CICSPlex SM API response codes.

api_response1_alt="*value*"
The CICS EIBRESP code or the CICSPlex SM API EYUDA response code as a text string. See RESP and RESP2 options for more information about CICS RESP values and EYUDA values for information about CICSPlex SM API response codes.

api_response2="*value*"
The CICS EIBRESP2 code or the CICSPlex SM API EYUDA reason code as a numeric value. The value for a successful request is 0. See RESP and RESP2 options for more information about CICS RESP2 values and EYUDA values for information about CICSPlex SM API reason codes.

api_response2_alt="*value*"
The CICS EIBRESP2 code or the CICSPlex SM API EYUDA reason code as a text string. See RESP and RESP2 options for more information about CICS RESP2 values and EYUDA values for information about CICSPlex SM API reason codes.

recordcount="*value*"
The total number of records returned by the request.

The **recordcount** attribute always indicates the total number of records in the result including those for which the request failed. If a record range is specified, the value of **recordcount** can be greater than the number of records returned by the request.

displayed_recordcount="*value*"
A count of the records returned by the request.

If a record range is specified, the value of **displayed_recordcount** can be less than the total number of records in the result as indicated by **recordcount**.

successcount="*value*"
For DELETE requests only, a count of the records for which the request succeeded.

cachetoken="*value*"
A token identifying a results cache containing a retained set of results. A **cachetoken** attribute is present only in the response to a GET request that specifies the NODISCARD option. The value of **cachetoken** is a 16–character, fixed-length hexadecimal representation of a store clock value.

Examples

The following example shows the result of a successful DELETE request:

```
<resultsummary api_response1="1024" api_response1_alt="OK"
api_response2="0" api_response2_alt="" recordcount="20"
dispayed_recordcount="20" successcount="18" />
```

The following example shows the result of a DELETE request that does not complete successfully:

```
<resultsummary api_source="CICSPlex SM" api_function="SET"  api_response1="1038"
api_response1_alt="TABLEERROR" api_response2="1361" api_response2_alt="DATAERROR"
recordcount="0" dispayed_recordcount="0" successcount="0"/>
```

The following example shows the result of a GET request that includes the NODISCARD option:

```
<resultsummary api_response1="1024" api_response1_alt="OK"
api_response2="0" api_response2_alt="" recordcount="2"
dispayed_recordcount="2" cachetoken="C36E880ECAC5818B" />
```

<records> element

The <records> element contains details of the records returned by a request. The <records> element contains one child element for each record returned by the request. If the request returns no records, the <records> element is absent.

Contained by:

"<response> element" on page 32

Contains:

The <records> element contains one child element for each instance of a resource changed by the request. These child elements are named by the resources identified in the request URI. For example, if a PUT request makes changes to CICS local files, the <records> element of the PUT response contains one or more child elements named <cicslocalfile>; that is, one <cicslocalfile> element for each record returned by the request. Child element names are derived from the CICS management client interface resource names. See "CMCI resource names" on page 17 for a complete list of these names.

Attributes

The <records> element has no attributes if its own.

The attributes of child elements specify the values of the attributes of the identified resource. All the attributes of the resource are present in the element, whether or not the value of the attribute has changed. Attributes with values that are left blank take their default values. See CICSPlex SM resource tables for a list of required attributes for each resource.

Each resource element also includes a **_keydata** attribute that takes the form _keydata="*data string*", where *data string* is a string of characters that uniquely identifies an instance of the resource.

Example

The following example shows a single record listing an instance of a CICSLocalFile resource.

```
<records>
    <cicslocalfile _keydata="C4C6C8C3E2C44040" accessmethod="VSAM" add="ADDABLE" addcnt="0"
    basedsname="USER1.ALLAPPL.DFHCSD" blockformat="BLOCKED" blockkeyln="N/A"
    blocksize="N/A" browse="BROWSABLE" browsecnt="2115" browupdcnt="0" datasettype="K"
    delete="DELETABLE" dexcpcnt="0" disposition="SHARE" dsname="USER1.ALLAPPL.DFHCSD"
    emptystatus="NOEMPTYREQ" enablestatus="ENABLED" exclusive="NOTAPPLIC" file="DFHCSD"
    fwdrecstatus="NOTFWDRCVBLE" getcnt="115" getupdcnt="0" gmtfilecls="00:00:00.0000"
    gmtfileopn="00:00:00.0000" iexcpcnt="0" journalnum="0" keylength="0"
    keyposition="0" locdelcnt="0" lsrpoolid="01" numactstring="0" numdatbuff="0"
    numindexbuff="0" numstringwt="0" object="BASE" openstatus="CLOSED"
    rbatype="NOTAPPLIC" read="READABLE" readinteg="NOTAPPLIC" recordformat="VARIABLE"
    recordsize="0" recovstatus="NOTRECOVABLE" reltype="NOTAPPLIC" rlsaccess="NOTRLS"
    rlsreqwtto="0" strings="6" timeclose="0:00:00.0" timeopen="0:00:00.0"
    update="UPDATABLE" updatecnt="0" vsamtype="NOTAPPLIC" wstrccurcnt="0" wstrcnt="0"
    eyu_cicsname="U1WUIA" eyu_cicsrel="E660" eyu_reserved="0"/>
</records>
```

<errors> element

The <errors> element contains diagnostic information associated with a CICS management client interface request. The response to a CICS management client interface request contains an <errors> element if there is diagnostic information to be displayed in any child elements.

The <errors> element contains one or more <feedback> child elements associated with the request. The <errors> element has no associated attributes.

Contained by:

"<response> element" on page 32

Contains:

"<feedback> element" on page 35

Examples

<feedback> element

The <feedback> element contains diagnostic data from a FEEDBACK record associated with a CICS management client interface request. The <errors> element can contain one or more <feedback> elements for each record returned by the request.

See Retrieving feedback records for more information about FEEDBACK records.

For a CICS management client interface PUT install request, a <feedback> element can contain the following child elements:

- One <installerror> element for each install action that fails with a BINSTERR record
- One <inconsistentscope> element for each install action that fails with a BINCONSC record
- One <inconsistentset> element for each install action that fails with a BINCONRS record

Contained by:

"<errors> element" on page 34

Contains:

"<installerror> element" on page 36
"<inconsistentscope> element" on page 37
"<inconsistentset> element" on page 38

Attributes

action="*action_name*"
The name of the action that has failed.

attribute1="*attribute_name*"
The name of one of up to six attributes associated with the error.

attribute2="*attribute_name*"
The name of one of up to six attributes associated with the error.

attribute3="*attribute_name*"
The name of one of up to six attributes associated with the error.

attribute4="*attribute_name*"
The name of one of up to six attributes associated with the error.

attribute5="*attribute_name*"
The name of one of up to six attributes associated with the error.

attribute6="*attribute_name*"
The name of one of up to six attributes associated with the error.

eibfn="*eibfn_number*"
The function code associated with the request.

eibfn_alt="*function*"
The name of the function associated with the request.

errorcode="*error_code*"
The CICSPlex SM error code associated with the resource.

eyu_cicsname="*name*"
The name of the CICS region or CICSplex associated with the error.

keydata="*data*"
A string of data that identifies the instance of a resource associated with the error.

resp="*resp_number*"
The CICS RESP code or the CICSPlex SM API EYUDA response code as a numeric value.

resp2="*resp2_number*"
The CICS RESP2 code or the CICSPlex SM API EYUDA reason code as a numeric value.

resp_alt="*resp_alt_text*"
The text equivalent for the resp value. For example, the text equivalent of a resp value of 16 is INVREQ.

Example

```
<feedback action="STOP" eibfn="4C10" eibfn_alt="DISCARD FILE" eyu_cicsname="REG1"
keydata="FILEX" resp="16" resp_alt="INVREQ" resp2="2"/>
```

<installerror> element

The <installerror> element contains diagnostic data from a BINSTERR record associated with a CICS management client interface PUT install request. The <installerror> element is contained by the <feedback> element.

See Evaluating BINSTERR resource table records for more information about BINSTERR records.

Contained by:

"<feedback> element" on page 35

Attributes

eibfn="*eibfn_number*"
The function code associated with the request.

eyu_cicsname="*name*"
The name of the CICS region or CICSplex associated with the installation error.

cresp1="*resp_number*"
The CICS RESP code or the CICSPlex SM API EYUDA response code as a numeric value.

cresp2="*resp2_number*"
The CICS RESP2 code or the CICSPlex SM API EYUDA reason code as a numeric value.

errorcode="*error_code*"
The CICSPlex SM error code associated with the resource.

ressname="*resource_name*"
The name of the resource associated with the error.

resver="*resource_version*"
The version number of the resource associated with the error.

Example

```
<installerror eibfn="3036" errorcode="4" eyu_cicsname="MYWUIB" resourcename="URIMAP_X"
resourceversion="1" resp="16" resp2="626"/>
```

<inconsistentscope> element

The <inconsistentscope> element contains diagnostic data from a BINCONSC record associated with a CICS management client interface PUT request. The <inconsistentscope> element is contained by the <feedback> element.

See Evaluating BINCONSC resource table records for more information about BINCONSC records.

Contained by:

"<feedback> element" on page 35

Attributes

eibfn="*eibfn_number*"
The function code associated with the request.

eyu_cicsname="*name*"
The name of the CICS region or CICSplex associated with the inconsistent scope error.

erroroperation="*value*"
A numeric value that identifies the operation being performed when the error occurred

errorcode="*error_code*"
The CICSPlex SM error code associated with the resource.

targetassignment="*assignment*"
The assignment for the target scope.

targetdescription="*description*"
The resource description for the target scope.

relatedassignment="*assignment*"
The resource assignment for the related scope.

relateddescription="*assignment*"
The resource description for the related scope.

relatedscope="*scope*"
The name of the related scope.

Example

An attempt to add a resource assignment to a resource description fails because the target scope and related scope of the resource assignment are the same producing error code 1 for BINCONSC.

```
<?xml version="1.0"?>
<response xmlns="http://www.example.com/xmlns/prod/CICS/smw2int"
xmlns:xsi="http://www.example.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.example.com/xmlns/prod/CICS/smw2int
http://example.com:27870/CICSSystemManagement/schema/CICSSystemManagement.xsd"
version="2.0" connect_version="0420">
    <resultsummary api_function="PERFORM SET" api_response1="1038" api_response2="1361"
    api_response1_alt="TABLEERROR" api_response2_alt="DATAERROR" recordcount="1" />
    <errors>
        <feedback action="ADDTODSC">
            <inconsistentscope erroroperation="1" errorcode="1" targetassignment="ATEST1"
         targetscope="MCLMASA" relatedassignment="ATEST1" relatedscope="MCLMASA" />
        </feedback>
        <feedback keydata="C1E3C5E2E3F14040" action="ADDTODSC" errorcode="24"
       attribute1="RESASSGN" />
    </errors>
</response>
```

<inconsistentset> element

The <inconsistentset> element contains diagnostic data from a BINCONRS record associated with a CICS management client interface PUT request. The <inconsistentset> element is contained by the <feedback> element.

See Evaluating BINCONRS resource table records for more information about BINCONRS records.

Contained by:

"<feedback> element" on page 35

Attributes

candidatename="*resource_name*"
The name of the candidate resource.

candidateversion="*resource_version*"
The version number of the candidate resource.

candidategroup="*group_name*"
The resource group of the candidate resource.

candidateassignment="*assignment*"
The assignment of the candidate resource.

candidatedescription="*assignment*"
The description of the candidate resource.

candidateusage="*usage*"
The assignment usage of the candidate resource.

candidatesystemgroup="*group_name*"
The system group of the candidate resource.

candidatetype="*system_type*"
The system type of the candidate resource.

candidateoverride="*assignment_override*"
The assignment override of the candidate resource.

eyu_cicsname="*name*"
The name of the CICS region associated with the installation error.

erroroperation="*value*"
A numeric value that identifies that the operation being performed when the error occurred

existingname="*resource_name*"
The name of the existing resource.

existingversion="*resource_version*"
The version number of the existing resource.

existinggroup="*group_name*"
The resource group of the existing resource.

existingassignment="*assignment*"
The assignment of the existing resource.

existingdescription="*assignment*"
The description of the existing resource.

existingusage="*usage*"
The assignment usage of the existing resource.

existingsystemgroup="*group_name*"
The system group of the existing resource.

existingtype="*system_type*"
The system type of the existing resource.

existingoverride="*assignment_override*"
The assignment override of the existing resource.

Example

An attempt to add a resource assignment to a resource description fails because of a conflict with an existing resource assignment associated with the resource description.

```
<?xml version="1.0"?>
<response xmlns="http://www.example.com/xmlns/prod/CICS/smw2int"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.example.com/xmlns/prod/CICS/smw2int
 http://example.com:27870/CICSSystemManagement/schema/CICSSystemManagement.xsd"
 version="2.0" connect_version="0420" >
    <resultsummary api_function="PERFORM SET" api_response1="1038" api_response2="1361"
    api_response1_alt="TABLEERROR" api_response2_alt="DATAERROR" recordcount="1" />
    <errors >
        <feedback action="ADDTODSC">
            <inconsistentset candidatename="ATEST1" candidateversion="1" candidategroup="ATEST1"
         candidateassignment="ATEST2" candidatedescription="ATEST" candidateusage="LOCAL"
         candidatetype="TARGET" candidateoverride="NO" eyu_cicsname="MCLMASA" erroroperation="4"
         existingname="ATEST1" existingversion="1" existinggroup="ATEST1" existingassignment="ATEST1"
         existingdescription="ATEST" existingusage="REMOTE" existingtype="TARGET"
existingoverride="NO"  />
            <inconsistentset candidatename="ATEST2" candidateversion="1" candidategroup="ATEST1"
         candidateassignment="ATEST2" candidatedescription="ATEST" candidateusage="LOCAL"
         candidatetype="TARGET" candidateoverride="NO" eyu_cicsname="MCLMASA" erroroperation="4"
         existingname="ATEST2" existingversion="1" existinggroup="ATEST1" existingassignment="ATEST1"
         existingdescription="ATEST" existingusage="REMOTE" existingtype="TARGET"
existingoverride="NO"  />
        </feedback >
        <feedback keydata="C1E3C5E2E3F24040" action="ADDTODSC" errorcode="13" attribute1="RESASSGN"  />
    </errors>
</response>
```

CMCI escape sequences

Certain characters are restricted in CMCI URIs. For these characters, you must substitute an escape sequence consisting of the percent character (%) followed by a hexadecimal value.

The following table lists the restricted characters and their hexadecimal escape sequences.

Table 5. Restricted characters and their escape sequences

Character	Escape sequence	Character	Escape sequence
SPACE	%20	. (period)	%2E
!	%21	/	%2F
#	%23	: (colon)	%3A
$	%24	; (semicolon)	%3B
%	%25	<	%3C
&	%26	=	%3D
' (single quote)	%27	>	%3E
(	%28	?	%3F
)	%29	@	%40
*	%2A	[	%5B
+	%2B	]	%5D
, (comma)	%2C	¬ (not)	%AC
- (minus sign)	%2D		

Examples

The following example shows a criteria string from a CICS management client interface URI before and after encoding with escape characters:

Before:

```
CRITERIA=(TRANID=P* AND PROGRAM=PAY* AND STATUS=ENABLED) AND
((USECOUNT>0 AND STGVCNT>0) OR NOT RESTARTCNT=0)
```

After:

```
CRITERIA=%28TRANID%3DP%2A%20AND%20PROGRAM%3DPAY%2A%20AND%20STATUS%3DENABLED%29%20AND%20
%28%28USECOUNT%3E0%20AND%20STGVCNT%3E0%29%20OR%20NOT%20RESTARTCNT%3D0%29
```

The following example shows a parameter string from a CICS management client interface URI before and after encoding with escape characters:

Before:

```
PARAMETER=STARTDATE(07/17/2006) STARTTIME(17:00) INTERVAL(300)
```

After:

```
PARAMETER=STARTDATE%2807%2F17%2F2006%29%20STARTTIME%2817%3A00%29%20INTERVAL%28300%29
```

The interface also supports an optional period (%2E) at the end of criteria and parameter strings.

CMCI problem determination

The CICS management client interface (CMCI) provides return codes and several types of diagnostic information to help you determine the cause of errors.

If the CICS management client interface receives a request from the client that is not valid, the request fails and the API returns a non-OK code in the HTTP header of the response. The CICS management client interface does not process the request and provides no other information. See “CICS management client interface error messages” on page 46 for a description of these response codes.

If the CICS management client interface determines that the request is valid, it returns an HTTP 200 OK response code and generates an EXEC CICS or an EXEC CPSM call. If this call encounters any errors, the API returns diagnostic information in the XML body of the response. This information takes the form of response and reason codes in the <resultsummary> element of the XML response, and in certain cases, feedback records that provide extra information.

The <resultsummary> element contains four attributes that you can use to determine the cause of a failure:

- api_function
- api_response1
- api_response1_alt
- api_response2
- api_response2_alt

See “<resultsummary> element” on page 32 for a description of these attributes.

If the CICS management client interface receives any exception conditions, it passes them to the client in the <errors> element of the XML response. The <errors> element is absent if the request completes successfully. The <errors> element contains one or more <feedback> elements each containing a FEEDBACK record. In the case of certain BAS errors, the <feedback> element contains an additional <installerror>, <inconsistentscope>, or <inconsistentset> element containing a BINSTERR, BINCONSC, or BINCONRS record respectively.

Example: Diagnosing a CICS management client interface error using <resultsummary> information

The CICS management client interface, <resultsummary> element contains return codes that help you determine the cause of errors.

About this task

You have attempted to perform a START action on a CICSManagedRegion resource (that is, a CICSPlex SM managed CICS region or MAS) using a CICS management client interface PUT request with the following URI and XML body:

```
http://wmvs2c.mycorp.com:27650/CICSSystemManagement/CICSManagedRegion/IBEUR912?
CRITERIA=CICSNAME%3DDEVRGN1

    <request>
          <action name="START"/>
    </request>
```

Although you receive an HTTP/1.1 200 OK response from the CICS management client interface, no record is returned and the response includes the following <resultsummary> information:

```
<resultsummary api_function="PERFORM SET" api_response1="1026" api_response1_alt="NOTFOUND"
api_response2="1301" api_response2_alt="ACTION" recordcount="1"/>
```

To understand the reason for this problem:

Procedure

1. Determine the source of the error from the **api_function** attribute.

   This attribute indicates that the CICS management client interface has attempted to call the PERFORM SET function but the call has failed.
2. Determine the reason for the failure from the **api_response1_alt** and **api_response2_alt** attributes.

   These values are CICSPlex SM EYUDA response and reason codes.

   The EYUDA response has the symbolic name NOTFOUND, and the EYUDA reason value has the symbolic name ACTION. These values mean that the attempted action was not found in the CICSPlex SM MAS resource table. The only actions available for the MAS resource table are STOP, STOPUNCON, and FORCEDISCON. The START action is not permitted. See the MAS table to confirm the actions available for a MAS resource.

Example: Diagnosing a CICS management client interface error using <feedback> information

The CICS management client interface provides FEEDBACK records in the <feedback> element of a CICS management client interface response. You can use the extra information provided in FEEDBACK records to diagnose problems with CICS management client interface requests.

About this task

You have attempted to delete the CICS local file resource MYFILE_A, using a client API DELETE request with the following URI:

```
http://mvs2c.mycorp.com:27850/CICSSystemManagement/CICSLocalFile/IBEUR912?
CRITERIA=file%3DMYFILE_A
```

Although you receive an HTTP/1.1 200 OK response from the CICS management client interface indicating that the request was correctly formed, no record is returned and the response includes the following information:

```
<resultsummary api_function="PERFORM SET" api_response1="1038" api_response1_alt="TABLEERROR"
api_response2="1361" api_response2_alt="DATAERROR" recordcount="1"/>
  <errors>
        <feedback action="DISCARD" eibfn="4C10" eibfn_alt="DISCARD FILE" eyu_cicsname="MAS1"
keydata="MYFILE_A"
     resp="16" resp2="2"/>
  </errors>
```

To understand the reason for this problem:

Procedure

1. Determine the source of the error from the **api_function** attribute.

   This attribute indicates that the CICS management client interface has attempted to call the PERFORM SET function but this has failed.
2. Determine the reason for the failure from the **api_response1_alt** and **api_response2_alt** attributes.

   These values are CICSPlex SM EYUDA response and reason codes.

   The EYUDA response value is TABLEERROR, and the EYUDA reason value is DATAERROR.
3. Determine the API call that failed by combining the information in the **api_function** attribute of the <resultsummary> tag with the **action** attribute from the <feedback> element.

   Using this information, you can determine that the failure occurred when the CICS management client interface was trying to delete the file using the CICSPlex SM API command **EXEC CPSM PERFORM SET ACTION('DISCARD').**

4. Identify the associated EXEC CICS command from the **eibfn_alt** attribute of the <feedback> element.

   The **eibfn_alt** value represents the command **EXEC CICS DISCARD FILE**.

5. Use the **resp** and **resp2** attributes of the <feedback> element to determine the exact cause of the failure.

   Both the values are presented in decimal format.

   a) Look up the symbolic name for the CICS **resp** value 16.

      See EIB fields for a list of EIBREP values.

      From this list, you can determine that an EIBRESP field with a value of 16 has the symbolic name INVREQ.

   b) Finally, look up the meaning of the INVREQ response with a **resp2** value of 2 for the **EXEC CICS DISCARD FILE** command.

      See DISCARD FILE for a list of RESP2 values for the INVREQ condition.

      From this list you can determine that the cause of the failure of an **EXEC CICS DISCARD FILE** command that returns an INVREQ condition with a RESP2 value of 2 is "The file is not closed".

Example: Diagnosing a CICS management client interface install error

To install a BAS resource using the CICS management client interface, you use a PUT request with an INSTALL action. If the installation fails, CICSPlex SM generates a BINSTERR record containing diagnostic information. This record is returned in an <installerror> element, which is a child element of <feedback>.

About this task

You have attempted to install a URIMAP resource named URIMAP_A into the region DEVRGN1, using a PUT request with the following URI and XML body:

```
http://wmvs2c.mycorp.com:27640/CICSSystemManagement/CICSDefinitionURIMap/IBEUR912?
CRITERIA=name%3DURIMAP_A

<request>
  <action name="INSTALL">
    <parameter name="TARGET" value="DEVRGN1"/>
    <parameter name="USAGE" value="LOCAL"/>
  </action>
</request>
```

Although you receive an HTTP/1.1 200 OK response from the CICS management client interface indicating that the request was correctly formed, the URIMAP is not installed. The response includes the following information:

```
<resultsummary api_function="PERFORM SET" api_response1="1038" api_response1_alt="TABLEERROR"
api_response2="1361" api_response2_alt="DATAERROR"  recordcount="1"/>
  <errors>
    <feedback action="INSTALL" attribute1="NAME" errorcode="31">  <installerror eibfn="3036"
        eibfn_alt="CREATE  URIMAP" errorcode="4" eyu_cicsname="IBWUIA" resourcename="URIMAP_A"
        resourceversion="1" resp="16" resp2="500"/>
    </feedback>
  </errors>
```

To understand the reason for this problem, perform these steps:

Procedure

1. Determine the source of the error from the **api_function** attributes.

   This attribute indicates that the CICS management client interface has tried to call the PERFORM SET function but this has failed.

2. Determine the reason for the failure from the **api_response1_alt** and **api_response2_alt** attributes.

   These values are CICSPlex SM EYUDA response and reason codes.

The EYUDA response value is TABLEERROR, and the EYUDA reason value is DATAERROR.

3. Determine the API call that failed by combining the information in the **api_function** attribute of the <resultsummary> tag with the **action** attribute from the <feedback> element.

   Using this information you can determine that the failure occurred when the CICS management client interface was trying to perform the CICSPlex SM API command **EXEC CPSM PERFORM SET ACTION('INSTALL')**
4. Identify the associated EXEC CICS command from the **eibfn_alt** attribute of the <installerror> element.

   The **eibfn_alt** represents the command **EXEC CICS CREATE URIMAP**
5. Use the value of the **errorcode** attribute of the <installerror> element to determine the CICSPlex SM BINSTERR error code.

   See BINSTERR Resource Table for a list of error codes and their meanings. In this case the **errorcode** value 4 is `(INSTFAIL) - Install failure`.
6. Use the **resp** and **resp2** attributes of the <installerror> element to determine the exact cause of the failure.

   Both the values are presented in decimal format.

   a) Look up the symbolic name for the CICS **resp** value 16.

      See EIB fields for a list of EIBRESP values.

      From this list, you can determine that an EIBRESP field with a value of 16 has the symbolic name INVREQ.

   b) Finally, look up the meaning of the INVREQ response with a **resp2** value of 500 for the **EXEC CICS CREATE URIMAP** commands.

      See RESP2 values for CREATE and CSD commands for a list RESP2 values for the **EXEC CICS CREATE** command.

CICS management client interface error messages

The CICS management client interface (CMCI) issues XML messages with message identifiers in the range DFHWU4001 to DFHWU5002 to indicate the status of CMCI requests. These messages are associated with HTTP response codes and are accompanied by explanations and error information to help you understand the cause of any failures.

A typical example of a CMCI message is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<error message_id="DFHWU4003" connect_version="0410">
<title>400 CICS management client interface HTTP Error</title>
<short>An error has occurred in the CICS management client interface. The request cannot be
processed.</short>
<full> An unknown query parameter was specified in the URI.</full>
<errorInfo name="ccc">ccc</errorInfo>
</error>
```

The message consists of the message identifier, title, short and long explanations, and specific error information. In this case the information in the <errorInfo> element identifies the unknown query parameter in the URI.

DFHWU4001 - DFHWU4029, HTTP 400 errors:

An HTTP 400 response code indicates that the request was not understood by the server. All HTTP 400 responses include the following information:

400 CICS management client interface Error

An error has occurred in the CICS management client interface. The request cannot be processed. *full explanation*

Where *full explanation* is one of the following:

- The resource name was not specified in the URI.

- The URI specified contains a PATH that exceeds the maximum allowable length of 256 bytes.
- An unknown query parameter was specified in the URI.
- The context was missing from the URI.
- The result cache token was missing from the URI.
- The body of the HTTP request was missing.
- The body of the HTTP request was not specified correctly.
- The record index specified in the URI is not valid.
- The record count specified in the URI is not valid.
- The record index was specified for a noncached result.
- Extraneous data was detected at the end of the URI.
- Multiple CRITERIA expressions were found in the URI.
- Multiple PARAMETER expressions were found in the URI.
- Multiple NODISCARD expressions were found in the URI.
- Multiple SUMMONLY expressions were found in the URI.
- CRITERIA is not valid for result cache operations.
- PARAMETER is not valid for result cache operations.
- The resource name was missing from the URI.
- A specified attribute was not valid for this resource.
- The DEFVER attribute was not specified or was specified with a value of zero.
- A value of a specified attribute was out-of-range or not valid.
- NODISCARD is only valid for HTTP GET requests.
- CRITERIA is not valid for HTTP POST requests.
- PARAMETER is not valid for HTTP POST requests.
- The result cache token specified exceeded its maximum allowable length.
- An action was specified in the HTTP body that was not valid.

DFHWU4030, HTTP 401 error:

401 Basic Authentication Error

Authentication with the server has failed.
A user name, or password, or both are required but missing or incorrect.

DFHWU4300, HTTP 403 error:

403 CICS management client interface Access Forbidden

An error has occurred in the CICS management client interface. Access to the specified result cache has been denied.
The result cache token specified in the URI does not belong to the user who made the request.

DFHWU4400 - DFHWU4402, HTTP 404 errors:

404 CICS management client interface Resource Not Found

An error has occurred in the CICS management client interface. The resource specified in the URI could not be found.
The resource type is not supported by this version of the CICS management client interface.

404 CICS management client interface Result Cache Record Not Found

An error has occurred in the CICS management client interface. The result cache record specified could not be found.
The result cache record index was out of range.

404 CICS management client interface Result Cache Not Found

An error has occurred in the CICS management client interface. The result cache specified could not be found.

The result cache token could not be found.

DFHWU4500, HTTP 405 errors:

405 CICS management client interface Method (*DELETE*|*GET*|*POST*|*PUT*|*HEAD*|*Unknown*) Not Allowed

An error has occurred in the CICS management client interface. The specified HTTP method is not allowed for the URI.

A method has been specified that is not valid for the URI sent to the CICS management client interface.

DFHWU5000 - DFHWU5002, HTTP 500 errors:

500 System Management client Internal Server Error

An internal error has occurred in the CICS management client interface.

Contact your system administrator.

500 CICS management client interface Short On Storage

The CICS management client interface server has gone Short On Storage below the bar.

Contact your system administrator.

500 CICS management client interface Short On Storage

There was insufficient GCDSA storage available to complete the request.

Contact your system administrator.

Using CMCI to query system initialization parameters

You use a CMCI GET request operating on the CICSSystemParameter external resource to discover information about system initialization parameters and their overrides.

Before you begin

You must have a connection with the CMCI; for more information, see Setting up CMCI.

Ensure that the system initialization parameters to be retrieved were valid for the CICS region at CICS startup. The retrieval operation can behave inconsistently if invalid parameter values have been corrected from the console at startup. Some parameters will show the corrected values while others will display their original SIT values.

About this task

A CMCI request can retrieve system initialization table parameters, system initialization table overrides, or a combination of both.

In common with many other CICSPlex SM operations, you can control which CICS regions the retrieval operates on by specifying context and scope.

Complete the following steps to retrieve information about system initialization parameters and their overrides.

Procedure

1. Form a CMCI request using the HTTP GET method and specifying CICSSystemParameter as the resource name.
   a) Use the **context** and **scope** parameters to specify which CICS regions the request is to operate on.
   b) In the URI, use a parameter expression incorporating the parameters **PARMSRCE** and **PARMTYPE** to specify which parameters or overrides to retrieve.

      Both of these parameters are mandatory. For **PARMTYPE**, you must specify a value of SIT. For **PARMSRCE**, specify one of the following options:

      COMBINED
      Combination of the original system initialization table definitions and any applied parameter overrides

      CONSOLE
      Override parameters as specified at startup on the system console

      JCL
      Override parameters provided through a JCL EXEC PGM statement

      SYSIN
      Override parameters from the startup job stream defined in the SYSIN data set

      TABLE
      The original system initialization table values extracted from the DFHSIT*xx* load module

   For more information about GET requests, see CICS management client interface GET requests.
2. Submit the CMCI request.

   The parameters and override values are returned in the order of the KEYWORD attribute of the SYSPARM resource table irrespective of their order in the source, for example the SYSIN data set.

   This behavior can produce unexpected results for the override parameters combinations: **SPCTR** and **SPCTR***xx*, and **STNTR** and **STNTR***xx* because the order in which these pairs are specified is significant. For example, if you specify SPCTRAP=ALL followed by SPCTR=OFF, SPCTRAP tracing is not available

but if you specify SPCTR=OFF followed by SPCTRAP=ALL, SPCTRAP tracing is available. However a request against the SYSPARM resource always returns **SPCTR** before **SPCTR***xx*.

Example

- This request retrieves the values of the control tables suffixes set to for region REGION in the CICSplex MYPLEX.

```
/CICSSystemManagement/CICSSystemParameter/<MYPLEX>/<REGION>?PARAMETER=PARMSRCE(COMBINED)
%20PARMTYPE(SIT)&CRITERIA=KEYWORD%3D++T%20AND%20NOT%20KEYWORD%3DMXT
```

The request returns a result summary and a number of matching records, for example:

```
<resultsummary api_response1="1024" api_response2="0" api_response1_alt="OK" api_response2_alt=""
recordcount="10" displayed_recordcount="10" />
<records>
<cicssystemparameter _keydata="C3D3E340404040404040404040404040404040404040404040404040404040404000000001"
eyu_cicsname="WLW26W1" eyu_cicsrel="E660" eyu_reserved="0" keyword="CLT" segnum="1" segtot="1"
source="TABLE"
totallen="2" type="SIT" value="NO" valuelen="2" />
< cicssystemparameter
_keydata="C6C3E340404040404040404040404040404040404040404040404040404040404000000001"
eyu_cicsname="WLW26W1" eyu_cicsrel="E660" eyu_reserved="0" keyword="FCT" segnum="1" segtot="1"
source="TABLE"
totallen="2" type="SIT" value="NO" valuelen="2" />
<cicssystemparameter _keydata="D4C3E340404040404040404040404040404040404040404040404040404040404000000001"
eyu_cicsname="WLW26W1" eyu_cicsrel="E660" eyu_reserved="0" keyword="MCT" segnum="1" segtot="1"
source="TABLE"
totallen="2" type="SIT" value="NO" valuelen="2" />
<cicssystemparameter _keydata="D4E7E340404040404040404040404040404040404040404040404040404040404000000001"
eyu_cicsname="WLW26W1" eyu_cicsrel="E660" eyu_reserved="0" keyword="MXT" segnum="1" segtot="1"
source="SYSIN"
totallen="3" type="SIT" value="100" valuelen="3" />
<cicssystemparameter _keydata="D9E2E340404040404040404040404040404040404040404040404040404040404000000001"
eyu_cicsname="WLW26W1" eyu_cicsrel="E660" eyu_reserved="0" keyword="RST" segnum="1" segtot="1"
source="TABLE"
totallen="2" type="SIT" value="NO" valuelen="2" />
< cicssystemparameter
_keydata="E2C9E340404040404040404040404040404040404040404040404040404040404000000001"
eyu_cicsname="WLW26W1" eyu_cicsrel="E660" eyu_reserved="0" keyword="SIT" segnum="1" segtot="1"
source="SYSIN"
totallen="2" type="SIT" value="EU" valuelen="2" />
<cicssystemparameter _keydata="E2D9E340404040404040404040404040404040404040404040404040404040404000000001"
eyu_cicsname="WLW26W1" eyu_cicsrel="E660" eyu_reserved="0" keyword="SRT" segnum="1" segtot="1"
source="TABLE"
totallen="2" type="SIT" value="1$" valuelen="2" />
<cicssystemparameter _keydata="E3C3E340404040404040404040404040404040404040404040404040404040404000000001"
eyu_cicsname="WLW26W1" eyu_cicsrel="E660" eyu_reserved="0" keyword="TCT" segnum="1" segtot="1"
source="TABLE"
totallen="2" type="SIT" value="DY" valuelen="2" />
<cicssystemparameter _keydata="E3E2E340404040404040404040404040404040404040404040404040404040404000000001"
eyu_cicsname="WLW26W1" eyu_cicsrel="E660" eyu_reserved="0" keyword="TST" segnum="1" segtot="1"
source="TABLE"
totallen="2" type="SIT" value="NO" valuelen="2" />
<cicssystemparameter _keydata="E7D3E340404040404040404040404040404040404040404040404040404040404000000001"
eyu_cicsname="WLW26W1" eyu_cicsrel="E660" eyu_reserved="0" keyword="XLT" segnum="1" segtot="1"
source="TABLE"
totallen="2" type="SIT" value="NO" valuelen="2" />
</records>
</response>
```

- The following request retrieves the values set for the **MXT** (MAX TASKS) system initialization parameter for all the regions in the CICsplex MYPLEX.

```
/CICSSystemManagement/CICSSystemParameter/<MYPLEX>?PARAMETER=PARMSRCE(COMBINED)
%20PARMTYPE(SIT)&CRITERIA=KEYWORD%3DMXT
```

- The following retrieves the values of the control tables suffixes set to for region REGION in the CICSplex MYPLEX.

```
/CICSSystemManagement/CICSSystemParameter/<MYPLEX>/<REGION>?PARAMETER=PARMSRCE(COMBINED)
%20PARMTYPE(SIT)&CRITERIA=KEYWORD%3D++T%20AND%20NOT%20KEYWORD%3DMXT
```

- The following request retrieves the names of the regions in CICSplex MYPLEX that have turned off the default runaway task time interval.

```
/CICSSystemManagement/CICSSystemParameter/<MYPLEX>?PARAMETER=PARMSRCE(COMBINED)
%20PARMTYPE(SIT)&CRITERIA=KEYWORD%3DICVR%20AND%20VALUE%3D0
```

- The following request retrieves the overrides applied to the region REGION in the CICSplex MYPLEX by the SYSIN override.

```
/CICSSystemManagement/CICSSystemParameter/<MYPLEX>/<REGION>?PARAMETER=PARMSRCE(SYSIN)%20PARMTYPE(SIT)
```

- The following request retrieves the overrides applied to the region REGION in the CICSplex MYPLEX by the JCL override.

```
/CICSSystemManagement/CICSSystemParameter/<MYPLEX>/<REGION>?PARAMETER=PARMSRCE(JCL)%20PARMTYPE(SIT)
```

Interacting with resource definitions

Get (retrieve), create, update, delete, and perform actions on BAS and CSD resource definitions using the CICS Management Client Interface (CMCI).

About this task

You can interact with resource definitions from CICSPlex SM in the following ways:

- Using the CMCI, which accepts HTTP requests from your application
- Using the CICS Explorer, which takes advantage of the CMCI
- Using the CICSPlex SM Web User Interface (WUI)
- Adding CICSPlex SM API commands directly to your application

Using CMCI to get a resource definition

Get or retrieve a BAS or CSD resource definition using the CICS Management Client Interface (CMCI).

Before you begin

You must have a connection with the CMCI. For more information, see Setting up CMCI.

About this task

Use a GET request to retrieve a resource definition; specify the URI for the request, following the instructions in “CMCI GET requests” on page 5 and “CMCI GET request URI” on page 7.

Note:

- If you are requesting a BAS resource, the *SCOPE* parameter, if specified in the URI, is ignored.
- If you are requesting a CSD resource, you must specify the *SCOPE* and PARAMETER=CSDGROUP(*value*) in the URI.

  The *SCOPE* must be the CICS system from which you are retrieving the resource. The **CSDGROUP** parameter is passed to instruct CICSPlex SM to search in the CSD groups in the specified system.

Example

The following example shows how to retrieve a BAS transaction definition named TRN1.

```
GET http://exampledomain.com:12345/CICSSystemManagement/
    CICSDefinitionTransaction/<CONTEXT>/
    ?CRITERIA=NAME%3DTRN1
```

The following example shows a GET request to retrieve a CSD transaction definition named TRN2 that is installed in a CICS region named MYREGION. Note that SCOPE specifies MYREGION and that CSDGROUP(*) is specified in the URI.

```
GET http://exampledomain.com:12345/CICSSystemManagement/
    CICSDefinitionTransaction/<CONTEXT>/MYREGION/
    ?CRITERIA=NAME%3DTRN2&PARAMETER=CSDGROUP%28%2A%29
```

Using CMCI to create a resource definition

Create a BAS or CSD resource definition using the CICS Management Client Interface (CMCI).

Before you begin

You must have a connection with the CMCI. For more information, see Setting up CMCI.

About this task

Use a POST request to create a resource definition. For the request, specify the URI and an XML body containing the parameters and attributes for the resource, as instructed in the **Procedure** below.

For information about POST requests, see "CMCI POST requests" on page 11.

Procedure

1. Specify the URI for the request, following the instructions in "CMCI POST request URI" on page 12.

   Note:

   - If you are creating a BAS resource, the *SCOPE* parameter, if specified in the URI, is ignored.
   - If you are creating a CSD resource, specify the *SCOPE* in the URI. The *SCOPE* must be the CICS system in which you are creating the resource.
2. Add an XML body containing the parameters and attributes for the resource.

   The XML body comprises a <create> element inside a <request> element. Specify the attributes of the new resource in the <create> element.

   For information about CMCI XML elements, see "CMCI XML" on page 27.

   Note:

   - If you are creating a BAS resource, you must include a `defver` attribute in the XML body.
   - If you are creating a CSD resource, you must specify the CSD parameter and a `csdgroup` attribute in the XML body. Do not include a `defver` attribute.

Example

The following example shows how to create a BAS transaction definition.

```
POST http://exampledomain.com:12345/CICSSystemManagement/
     CICSDefinitionTransaction/<CONTEXT>/

<request>
   <create>
      <attributes name="NAME" defver="1" program="test"/>
   </create>
</request>
```

This example shows how to create a CSD transaction definition. Note that the CSD parameter and the `csdgroup` attribute are added to the XML, and that the `defver` attribute is excluded.

```
POST http://exampledomain.com:12345/CICSSystemManagement/
     CICSDefinitionTransaction/<CONTEXT>/<SCOPE>/

<request>
   <create>
      <parameter name="CSD"/>
      <attributes name="resourceName" program="test" csdgroup="CSDGR"/>
   </create>
</request>
```

Both the examples, creating a BAS resource definition and creating a CSD resource definition, return a similar output when they complete successfully. A successful create returns a CICSPlex SM API response message that contains `api_response1="1024"` and `api_response1_alt="OK"` as shown below:

```
<response xmlns:...>
   <resultsummary api_response1="1024" api_response1_alt="OK".. />
   <records>
         ...
   </records>
</response>
```

Using CMCI to update a resource definition

Update a BAS or CSD resource definition using the CICS Management Client Interface (CMCI).

Before you begin

You must have a connection with the CMCI. For more information, see Setting up CMCI.

About this task

Use a PUT request to update a resource definition. For the request, specify the URI and an XML body containing the parameters and attributes for the resource, as instructed in the **Procedure** below.

For information about PUT requests, see “CMCI PUT requests” on page 13.

Procedure

1. Specify the URI for the request, following the instructions in “CMCI PUT request URI” on page 15.

   Note:

   - If you are updating a BAS resource, the *SCOPE* parameter, if specified in the URI, is ignored.
   - If you are updating a CSD resource, you must specify the *SCOPE* and PARAMETER=CSDGROUP(*value*) in the URI.

     The *SCOPE* must be the CICS system in which you are updating the resource. The **CSDGROUP** parameter is passed to instruct CICSPlex SM to search in the CSD groups in the specified system.
2. Add an XML body containing the parameters and attributes for the resource.

   The XML body comprises an <update> element inside a <request> element. Specify new attribute values in the <update> element.

   For information about CMCI XML elements, see “CMCI XML” on page 27.

   Note:

   - If you are updating a BAS resource, you must include a `defver` attribute in the XML body.
   - If you are updating a CSD resource, do not include a `defver` attribute.

Example

The following example shows how to update a BAS transaction definition. In this example, the value of the program name for transaction definition TRN1 is updated to PRGM2.

```
PUT http://exampledomain.com:12345/CICSSystemManagement/
    CICSDefinitionTransaction/<CONTEXT>/
    ?CRITERIA=NAME%3TRN1

<request>
   <update>
      <attributes program="PRGM2"/>
   </update>
</request>
```

The following example shows an update request for a CSD transaction definition. In this example, a PUT request is issued to update the transaction named TRN1, in CSD group GRP1 in a CICS system named MYREGION. The program name of transaction TRN1 is updated to PRGM2. Note that SCOPE specifies MYREGION and that PARAMETER=CSDGROUP(GRP1) is specified in the URI. Note also that the `defver` attribute is excluded in the XML body.

```
PUT http://exampledomain.com:12345/CICSSystemManagement/
    CICSDefinitionTransaction/<CONTEXT>/MYREGION/
    ?CRITERIA=NAME%3DTRN1&PARAMETER=CSDGROUP%28GRP1%29

<request>
   <update>
      <attributes program="PRGM2"/>
   </update>
</request>
```

Both the examples, updating a resource definition and updating a CSD resource definition, return a similar output when they complete successfully. A successful update returns a CICSPlex SM API response message that contains `api_response1="1024"` and `api_response1_alt="OK"` as shown:

```
<response xmlns:...>
   <resultsummary api_response1="1024" api_response1_alt="OK".. />
   <records>
         ...
   </records>
</response>
```

Using CMCI to perform actions on a resource definition

Perform actions on a BAS or CSD resource definition using the CICS Management Client Interface (CMCI).

Before you begin

You must have a connection with the CMCI. For more information, see Setting up CMCI.

About this task

Use a PUT request to perform actions on a resource definition. For the request, specify the URI and an XML body containing the parameters and attributes for the resource, as instructed in the **Procedure** below.

For information about PUT requests, see “CMCI PUT requests” on page 13.

Procedure

1. Specify the URI for the request, following the instructions in “CMCI PUT request URI” on page 15.

   Note:

   - If you are performing actions on a BAS resource, the *SCOPE* parameter, if specified in the URI, is ignored.
   - If you are performing actions on a CSD resource, you must specify the *SCOPE* and PARAMETER=CSDGROUP(*value*) in the URI.

     The *SCOPE* must be the CICS system in which you are performing actions on the resource. The **CSDGROUP** parameter is passed to instruct CICSPlex SM to search in the CSD groups in the specified system.
2. Add an XML body containing the parameters and attributes for the resource. Specify the action that you want to perform.

   The XML body comprises an `<action name="action_name">` element inside a <request> element. If the action requires parameters, you must specify these in <parameter> elements inside the <action> element.

Each resource table includes a list of the actions that you can perform.

For information about CMCI XML elements, see “CMCI XML” on page 27.

Example

The following example shows an INSTALL request that is applied to all BAS transaction definitions in the scope that have a name starting with AA.

```
PUT http://exampledomain.com:12345/CICSSystemManagement/
    CICSDefinitionTransaction/<CONTEXT>/
    ?CRITERIA=NAME%3DAA*

<request>
   <action name="INSTALL">
      <parameter name="FORCEINS" value="NO"/>
      <parameter name="USAGE" value="LOCAL"/>
   </action>
</request>
```

The following example shows how to specify a CSDINSTALL action request. This request installs all CSD transactions that have a name starting with AA in CSD group GRP1 in a CICS region named MYREGION. Note that SCOPE specifies MYREGION and that PARAMETER=CSDGROUP(GRP1) is specified in the URI. The parameters that you can specify for an action on a CSD resource are listed in the command description.

```
PUT http://exampledomain.com:12345/CICSSystemManagement/
    CICSDefinitionTransaction/<CONTEXT>/MYREGION/
    ?CRITERIA=NAME%3DAA*&PARAMETER=CSDGROUP%28GRP1%29

<request>
   <action name="CSDINSTALL">
   </action>
</request>
```

Using CMCI to delete a resource definition

Delete a BAS or CSD resource definition using the CICS Management Client Interface (CMCI).

Before you begin

You must have a connection with the CMCI. For more information, see Setting up CMCI.

About this task

Use a DELETE request to delete a resource definition; specify the URI for the request, following the instructions in “CMCI DELETE requests” on page 3 and “CMCI DELETE request URI” on page 4.

If you want to delete a resource that has parameters, you must specify the parameters for the resource in the URI.

Note:

- If you are deleting a BAS resource, the *SCOPE* parameter, if specified in the URI, is ignored.
- If you are deleting a CSD resource, you must specify the *SCOPE* and PARAMETER=CSDGROUP(*value*) in the URI.

  The *SCOPE* must be the CICS system from which you are deleting the resource. The **CSDGROUP** parameter is passed to instruct CICSPlex SM to search in the CSD groups in the specified system.

Optionally, you can specify the details of your request in the XML body; however, these details in the XML body are not required to perform a delete request. The XML body comprises a <delete> element inside a <request> element. Not all clients can parse the XML body; for that reason, you must add the parameter to the URI to specify which resource to delete.

Example

The following example shows how to delete a BAS transaction definition. In this example, all transactions named AAAA are deleted.

```
DELETE http://exampledomain.com:12345/CICSSystemManagement/
       CICSDefinitionTransaction/?CRITERIA=NAME%3DAAAA
```

The following example shows how to delete a CSD transaction definition named TRN1 in CSD group GRP1 in a CICS system named MYREGION. A delete request does not require an XML body. Note that SCOPE specifies MYREGION and that PARAMETER=CSDGROUP(GRP1) is specified in the URI.

```
DELETE http://exampledomain.com:12345/CICSSystemManagement/
       CICSDefinitionTransaction/<CONTEXT>/MYREGION/
       ?CRITERIA=NAME%3DTRN1&PARAMETER=CSDGROUP%28GRP1%29
```

The following delete request for the CSD transaction TRN1 includes an XML body; it is equivalent to the previous request.

```
DELETE http://exampledomain.com:12345/CICSSystemManagement/
       CICSDefinitionTransaction/<CONTEXT>/MYREGION/
        ?CRITERIA=NAME%3DTRN1&PARAMETER=CSDGROUP%28GRP1%29

<request>
   <delete>
      <parameter name="CSD"/>
   </delete>
</request>
```

Notices

This information was developed for products and services offered in the United States of America. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property rights may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

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

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

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 jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

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

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

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who want to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119 Armonk,
NY 10504-1785
United States of America

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

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

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

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

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Programming interface information

IBM CICS supplies some documentation that can be considered to be Programming Interfaces, and some documentation that cannot be considered to be a Programming Interface.

Programming Interfaces that allow the customer to write programs to obtain the services of CICS Transaction Server for z/OS, Version 5 Release 6 (CICS TS 5.6) are included in the following sections of the online product documentation:

- Developing applications
- Developing system programs
- CICS TS security
- Developing for external interfaces
- Application development reference
- Reference: system programming
- Reference: connectivity

Information that is NOT intended to be used as a Programming Interface of CICS TS 5.6, but that might be misconstrued as Programming Interfaces, is included in the following sections of the online product documentation:

- Troubleshooting and support
- CICS TS diagnostics reference

If you access the CICS documentation in manuals in PDF format, Programming Interfaces that allow the customer to write programs to obtain the services of CICS TS 5.6 are included in the following manuals:

- Application Programming Guide and Application Programming Reference
- Business Transaction Services

- Customization Guide
- C++ OO Class Libraries
- Debugging Tools Interfaces Reference
- Distributed Transaction Programming Guide
- External Interfaces Guide
- Front End Programming Interface Guide
- IMS Database Control Guide
- Installation Guide
- Security Guide
- CICS Transactions
- CICSPlex System Manager (CICSPlex SM) Managing Workloads
- CICSPlex SM Managing Resource Usage
- CICSPlex SM Application Programming Guide and Application Programming Reference
- Java Applications in CICS

If you access the CICS documentation in manuals in PDF format, information that is NOT intended to be used as a Programming Interface of CICS TS 5.6, but that might be misconstrued as Programming Interfaces, is included in the following manuals:

- Data Areas
- Diagnosis Reference
- Problem Determination Guide
- CICSPlex SM Problem Determination Guide

Trademarks

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

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Apache, Apache Axis2, Apache Maven, Apache Ivy, the Apache Software Foundation (ASF) logo, and the ASF feather logo are trademarks of Apache Software Foundation.

Gradle and the Gradlephant logo are registered trademark of Gradle, Inc. and its subsidiaries in the United States and/or other countries.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

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

Red Hat®, and Hibernate® are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

Spring Boot is a trademark of Pivotal Software, Inc. in the United States and other countries.

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

Zowe™, the Zowe logo and the Open Mainframe Project™ are trademarks of The Linux Foundation.

The Stack Exchange name and logos are trademarks of Stack Exchange Inc.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability
These terms and conditions are in addition to any terms of use for the IBM website.

Personal use
You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use
You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights
Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM online privacy statement

IBM Software products, including software as a service solutions, (*Software Offerings*) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information (PII) is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect PII. If this Software Offering uses cookies to collect PII, specific information about this offering's use of cookies is set forth below:

For the CICSPlex SM Web User Interface (main interface):
Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user's user name and other PII for purposes of session management, authentication, enhanced user usability, or other usage tracking or functional purposes. These cookies cannot be disabled.

For the CICSPlex SM Web User Interface (data interface):
Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's user name and other PII for purposes of session management, authentication, or other usage tracking or functional purposes. These cookies cannot be disabled.

For the CICSPlex SM Web User Interface ("hello world" page):
Depending upon the configurations deployed, this Software Offering may use session cookies that do not collect PII. These cookies cannot be disabled.

For CICS Explorer:

Depending upon the configurations deployed, this Software Offering may use session and persistent preferences that collect each user's user name and password, for purposes of session management, authentication, and single sign-on configuration. These preferences cannot be disabled, although storing a user's password on disk in encrypted form can only be enabled by the user's explicit action to check a check box during sign-on.

If the configurations deployed for this Software Offering provide you, as customer, the ability to collect PII from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM Privacy Policy and IBM Online Privacy Statement, the section entitled *Cookies, Web Beacons and Other Technologies* and the IBM Software Products and Software-as-a-Service Privacy Statement.

Index

Special Characters

A

C

D

E

F

G

I

N

P

R

S

T

X