Control-M Workload Automation 9.0.00.200 Control-M/EM API Guide

May 2016

Contacting BMC Software

You can access the BMC Software website at http://www.bmc.com. From this website, you can obtain information about the company, its products, corporate offices, special events, and career opportunities.

United States and Canada

Address **BMC SOFTWARE INC**
2103 CITYWEST BLVD
HOUSTON TX
77042-2827
USA

Telephone ▪ **713 918 8800** ▪ **800 841 2031** **Fax** **713 918 8000**

Outside United States and Canada

Telephone **(01) 713 918 8800** **Fax** **(01) 713 918 8000**

For BMC Control-M Products that are licensed on the “per CPU – Full Capacity” unit of measurement and installed in an Amazon Web Services (“AWS”) or Microsoft Azure (“Azure”) cloud environment, a license is required for the total number of CPUs in each AWS or Azure instance upon which the Product is installed or which the Product manages, either remotely or locally. For AWS, one CPU is equivalent to one vCPU, as defined by AWS. For Azure, one CPU is equivalent to up to four Virtual Cores (as defined by Azure), rounded up to the closest multiple of four.

Restricted rights legend

Customer support

You can obtain technical support by using the BMC Software Customer Support website or by contacting Customer Support by telephone or e-mail. To expedite your inquiry, see "Before contacting BMC."

Support website

You can obtain technical support from BMC 24 hours a day, 7 days a week at http://www.bmc.com/support. From this website, you can:

- Read overviews about support services and programs that BMC offers
- Find the most current information about BMC products
- Search a database for issues similar to yours and possible solutions
- Order or download product documentation
- Download products and maintenance
- Report an issue or ask a question
- Subscribe to receive proactive e-mail alerts when new product notices are released
- Find worldwide BMC support center locations and contact information, including e-mail addresses, fax numbers, and telephone numbers

Support by telephone or e-mail

In the United States and Canada, if you need technical support and do not have access to the web, call 800 537 1813 or send an e-mail message to customer_support@bmc.com. (In the subject line, enter **SupID:<yourSupportContractID>**, such as SupID:12345). Outside the United States and Canada, contact your local support center for assistance.

Before contacting BMC

Have the following information available so that Customer Support can begin working on your issue immediately:

- Product information
  - Product name
  - Product version (release number)
  - License number and password (trial or permanent)
- Operating system and environment information
  - Machine type
  - Operating system type, version, and service pack or other maintenance level such as PUT or PTF
  - System hardware configuration

  - Serial numbers
  - Related software (database, application, and communication) including type, version, and service pack or maintenance level
- Sequence of events leading to the issue
- Commands and options that you used
- Messages received (and the time and date that you received them)
  - Product error messages
  - Messages from the operating system, such as `file system full`
  - Messages from related software

License key and password information

If you have questions about your license key or password, contact BMC as follows:

- (USA or Canada) Contact the Order Services Password Team at 800 841 2031, or send an e-mail message to ContractsPasswordAdministration@bmc.com.
- (Europe, the Middle East, and Africa) Fax your questions to EMEA Contracts Administration at +31 20 354 8702, or send an e-mail message to password@bmc.com.
- (Asia-Pacific) Contact your BMC sales representative or your local BMC office.

Third party Software

For the provisions described in the BMC License Agreement and Order related to third party products or technologies included in the BMC Product, see https://docs.bmc.com/docs/display/workloadautomation/Control-M+Workload+Automation+Documentation and click **Third-party software (TPS).**

Contents

1

Introduction to Control-M/EM API

Control-M/Enterprise Manager (Control-M/EM) API is an open interface for external applications that enables you to use the capabilities of Control-M Workload Automation. It is a set of Java classes that allows you to send requests from your own applications to the Control-M Workload Automation. Installation includes a sample GUI client, which you can use to familiarize with the API before you develop your own custom client applications. It enables you to use Control-M/EM API, submit requests and get the responses. See Running sample GUI client (on page 8).

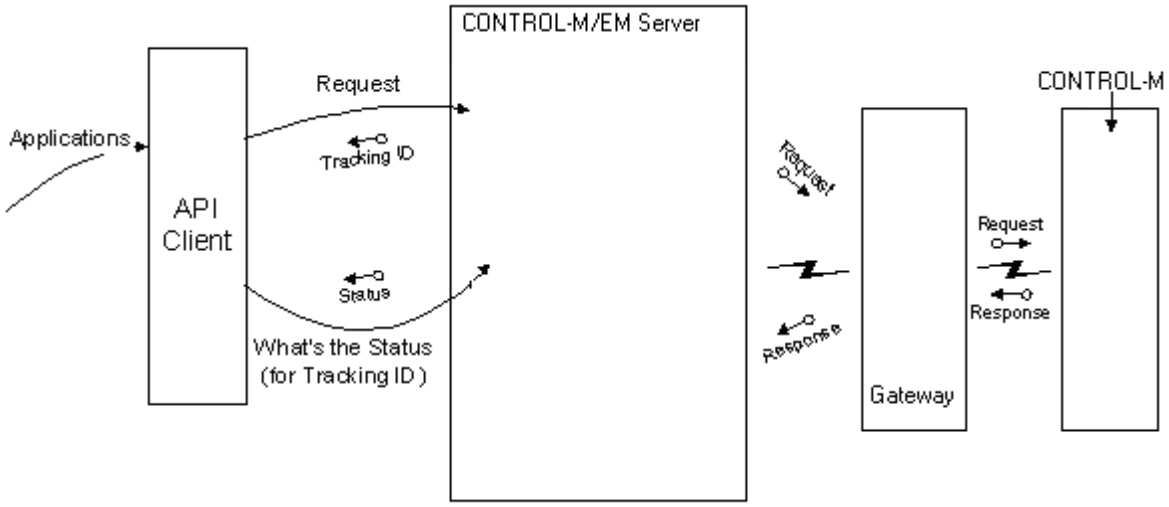
Control-M/EM API enables users of your application to perform various functions in the Control-M Business Integrated Scheduling environment, such as creating jobs and SMART Folders in Control-M Active Jobs, ordering jobs and SMART folders, performing job actions and changing Alert status. For practical examples, see Control-M/EM API flowchart (on page 7).

The following topics describe how to install, upgrade and uninstall Control-M/EM-API, send requests and receive responses from Control-M/EM, and error codes and exceptions.

- Control-M/EM API Installation (on page 10): Enables you install Control-M/EM API on Microsoft Windows and UNIX. This topic also includes a sample GUI client which you can use to familiarize with the API or with EMAPI before you develop your own custom client application.
- Control-M/EM API upgrade (on page 19): Enables you to upgrade from previous versions of Control-M/EM API.
- Control-M/EM API configuration (on page 13): Enables you to configure Control-M/EM API for use in your environment.
- Control-M/EM API Class references (on page 25): Defines class reference information about the functions that you can access when using Control-M/EM API in your project.
- Control-M/EM API requests (on page 35): Defines the different methods for creating, sending, and handling requests together with responses received from Control-M/EM.
- Request and response parameters (on page 46): Defines the Control-M/EM API request and response parameters. Requests are sent using the invoke (on page 31) method of EMXMLInvoker or EMBasicXMLInvoker.
- Diagnostics and troubleshooting (on page 187): Enables you to analyze the Control-M/EM API product's performance and prevent conflicts before they arise.
- Error codes and exceptions (on page 195): Defines the errors and exceptions, which may occur when the Control-M/Enterprise Manager API is being used.

Control-M/EM API flowchart

Control-M/EM API enables users of your application to submit requests to Control-M Workload Automation. The following figure shows the Control-M/EM API process flow.



Running sample GUI client

This procedure describes how to run a sample GUI client, which is included in Control-M/EM API installation. This enables you to familiarize with API before developing your own custom client application and allows you to submit requests and receive responses.

The sample GUI client is supplied as a source code and is located in **emapi-900\examples\EMAPIClient**.

➢ To run the sample GUI client:

1. Install and configure Control-M/EM API by completing the steps in Control-M/EM API Installation (on page 10) and Control-M/EM API configuration (on page 13).

   You should have access to an ANT installation.

2. Open command prompt window and change the directory to **emapi-900\examples\EMAPIClient.**
3. Copy the file **copy..\..\ctmemapi.properties** to the directory created in step 2.
4. Run the Control-M/EM API client by typing the following command:

   ```
   %ANT_HOME%/bin/ant run
   ```

   The GUI client appears.

5. Do one or more of the following:
   - Select either the EMXMLInvoker or EMBasicInvoker to initiate and send requests. For more information, see EMBasicXMLInvoker and EMXML Invoker class (on page 26).
   - Complete the following fields:
     - User
     - Response
     - OrderID
     - CTM

You can complete these fields or they can be completed when a relevant request is submitted (such as register_request).

- In the Request pane, type the XML request to submit. You can use the following variables in the XML request which is translated as a specified field in the window:
  - $USER_TOKEN$ - Defines user token for the request
  - $RESPONSE_TOKEN$
  - $CONTROL_M$ - Enables Control-M to submit the request
  - $ORDER_ID$ - Indicates job order id to submit the request

  For an example of XML request, see Sample GUI client XML variables (on page 9)

NOTE: You can use some predefined requests, such as Register, Unregister, Poll and Create_Simple_Job for distributed systems. To use one of these requests, select it and click **Invoke predefined.**

The predefine register request, uses emuser as user and empass as the password.

Sample GUI client XML variables

The following example describes the variables used in an XML request when using the Sample GUI client.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

  <SOAP-ENV:Body>

    <ctmem:request_add_condition
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

      <ctmem:user_token>$USER_TOKEN$</ctmem:user_token>

      <ctmem:control_m>$CONTROL_M$</ctmem:control_m>

      <ctmem:condition>

        <ctmem:name>COND0001</ctmem:name>

        <ctmem:odate>ODAT</ctmem:odate>

      </ctmem:condition>

    </ctmem:request_add_condition>

  </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

2

Control-M/EM API Installation

Control-M/EM API can be installed on Microsoft Windows and UNIX in any location from which you can connect using TCP/IP to the selected Control-M/EM installation. Control-M/EM API is supplied as a compressed file and when decompressed, this file creates a directory structure containing the Control-M/EM API files. You need to ensure that Control-M/EM API is compatible (on page 10) with your system.

The following procedures describe how to install, and configure Control-M/EM API:

- Installing Control-M/EM API (on page 11)
- Configuring Control-M/EM API (on page 13)

Compatibility

The following table describes the compatibility of Control-M/EM API with versions of Control-M/EM:

Component	Version
Control-M/EM	▪ 9.0.00 ▪ 8.0.00

The following table describes the compatibility of Control-M/EM API with versions of Control-M:

Component	Version
Control-M/Server for UNIX and Windows	▪ 9.0.00 ▪ 8.0.00 ▪ 7.0.00 ▪ 6.4.0x
Control-M for z/OS	▪ 9.0.00 ▪ 8.0.00 ▪ 7.0.00 ▪ 6.3.0x

Installing Control-M/EM API

This procedure describes how to install Control-M/EM API on Windows and UNIX.

Due to Java limitations, BMC Software recommends that you do not install Control-M/EM API in a directory with a path that contains spaces or other special characters. If you install Control-M/EM API on an account where an earlier version of Control-M/EM API is installed, see Control-M/EM API upgrade (on page 19) before continuing.

Before you begin

Ensure the following is installed:

- Control-M/EM in your network environment
- Java Developer's Kit (JDK) version 1.6.*x* or later or Java Runtime Environment (JRE) version 1.6.*x* or later on the computer hosting your project's working directory. **JAVA_HOME** environment variable should point to the JDK/JRE directory/library.

  NOTE: The **JAVA_HOME** environment refers to the directory where the JRE is installed. The JDK contains the JRE, but at a different level in the file hierarchy. For example, if the Java 2 SDK or JRE was installed in /ctm_em/user1, **JAVA_HOME** would be either:**/ctm_em/user1/jdk1.6.x/jre [JDK]** or **/ctm_em/user1/jre1.6.x [JRE].**

➢ To Install Control-M/EM API:

- Do one of the following to create the **emapi-900** directory:
  - **Microsoft Windows:** Unzip the **emapi-900-nt.zip** file from the **cdPath\Setup_files\TOOLS\EMAPI_FILES** directory on the Control-M/EM installation CD to any location.
  - **UNIX:**
    - **Solaris, AIX, HP-UX or HP-UX Itanium:** Locate the **emapi-900-UNIX.TAR.Z** tar file in the **cdPath/Setup_files/TOOLS/EMAPI_FILES** directory on the Control-M/EM installation CD and type the following command to open and uncompress the file to any directory:

      ```
      uncompress -c cdPath/TOOLS/EMAPI_FILES/emapi-900-UNIX.TAR.Z | tar xvf
      ```

    - **RedHat or Suse**: Locate the **emapi-900-UNIX.TAR.gz** tar file in the **cdPath/Setup_files/TOOLS/EMAPI_FILES** directory on the Control-M/EM installation CD and type the following command to open and uncompress the file to any directory:

      ```
      -gunzip -c cdPath/TOOLS/EMAPI_FILES/emapi-900-UNIX.TAR.gz | tar xvf-
      ```

  All Control-M/EM API files and sub-directories are located in this directory. See Control-M/EM API primary subdirectories (on page 12).

  NOTE: To uninstall Control-M/EM, delete the Control-M/EM API directory according to your version. For example, for version 9.0.00 delete the **emapi-900** directory. Do not copy the Control-M/EM API version 9.0.00 files directly over the previous installation This may cause unpredictable behavior.

Control-M/EM API primary subdirectories

The following table describes the Control-M/EM API primary subdirectories.

Directory	Description
emapi-900\etc	Contains the **jacorb.properties** file required for JacORB runtime (CORBA Java client library).
emapi-900\keystore	Contains the **emapi.keystore** demo file required for JacORB runtime when using SSL. For more information, see Key Store files.
emapi-900\classes	Contains the EMAPI libraries and third-party libraries.
emapi-900\xmldata	Contains the schema files that validate the XML formatted request strings sent and received using the API.
emapi-900\examples	Contains sample implementations of the AP and an example client. For more information, see Running sample GUI client (on page 8).

3

Control-M/EM API configuration

This topic describes how to configure Control-M/EM API for use in your project development environment.

Before using the Control-M/EM API in your project, you must initialize the API. You start and stop the Control-M/EM API by initializing the CORBA services. CORBA is an architecture and specification for creating, distributing, and managing distributed program objects in a network. In Control-M/EM API, CORBA is implemented through the use of JacORB, a third-party product that brokers requests among the components of distributed applications (acts as middleware).

API services start by using the EMXMLInvoker (on page 40) static method, `Init` (on page 29). When you finish using the Control-M/EM API use EMXMLInvoker static method, `done` (on page 27). It stops when the Control-M/EM API stops using the CORBA services. When your application finishes using the API, you can stop the API.

The Control-M/EM API uses services that need to be initialized and terminated only once. There is no need to initialize and stop for every new user session or call.

The following topics describe how to configure the Control-M/EM API, and prepare, write and run your project:

- Configuring Control-M/EM API (on page 13)
- Preparing the Control-M/EM API project environment (on page 15)
- Writing your project (on page 15)
- Running your project (on page 17)

Configuring Control-M/EM API

9his procedure describes how to configure Control-M/EM API.

The processes for configuring Control-M/EM API on computers running Microsoft Windows and on computers running UNIX are identical (except differences in the names of some of the files used, as described in Configuration files (on page 14)).

➢ To configure Control-M/EM API:

1. Locate Control-M/EM components and obtain the following information:
   - The CORBA Naming Service host name and port
   - Name of the Control-M/EM GUI Server
2. Change the current directory to the Control-M/EM API client's home directory, emapi-900.
3. Run one of the following files:
   - Microsoft Windows: **emapi-configure.bat**
   - UNIX: **emapi-configure**.
4. After running the script follow the on-screen prompts.

You are now ready to prepare your project and its environment to use the Control-M/EM API. See Control-M/EM API configuration (on page 13).

Configuration files

The following table describes the configuration files for Windows and UNIX:

Name (Windows)	Name (UNIX)	Description
emapi-configure.bat	emapi-configure	Configures Control-M/EM API: ▪ CORBA Naming Service ▪ Control-M/EM GUI Server to connect
emapi_env.bat	emapi_env.sh (users working in Korn Shell environment) emapi_env.csh (users working in C Shell and TC Shell environments)	Automatically adds API directory pathnames to the CLASSPATH environment variable.
emapi-admin.bat	emapi-admin	Starts an interactive utility for changing the names of the Control-M/EM GUI Server and updates the XMLDATAPATH property.
ctmemapi.properties		Contains the locations of: ▪ Control-M/EM GUI Server ▪ XMLDATAPATH property
NamingViewer.vbs	NamingViewer	Starts a Java GUI utility that displays the Naming Service repository graphically. For more information, see Configuring NamingViewer (browser for Naming Service).
changePass.bat	changePass	Encrypts the keystore password for demo certificates. For more information, see Key Store files.

Preparing the Control-M/EM API project environment

This procedure describes how to prepare your project environment. Prior to using the Control-M/EM API, you must configure the API for use in your project development environment. When you want to release your application for use, you must configure the Control-M/EM API running environment.

NOTE: Your project is in the development environment when you are creating and testing the project on your development computer. The running environment is the project environment when your application is released for use. Follow these steps for both the development environment and the running environment

➢ To prepare your project environment:

1. Set the environment variables for the environment by running the **emapi_env.bat** (or **emapi_env.sh**) file.
   - For **emapi_env.sh** on UNIX, run the following command:

     ```
     . emapi_env.sh
     ```

     Running the command with the above syntax ensures that the variables are valid after the command is run.
2. Copy the **emapi-900\ctmemapi.properties** file to your project's working directory.

Writing your project

This procedure describes how to use Control-M/EM API to write your project.

NOTE: You can use the Sample class (on page 16) as a basis for building your project.

➢ To write your project:

1. Import Control-M/EM API into your project by typing the following command:

   ```
   import com.bmc.ctmem.emapi.*;
   ```
2. Create a class that uses the Control-M/EM API functionality that you want to use in your project.
3. Start Control-M/EM API by using the EMXMLInvoker static method, Init (on page 29) by typing the following command:

   ```
   EMXMLInvoker.init();
   ```
4. Select the Control-M/EM GUI Server by typing the following command:

   ```
   ComponentType gsr_comp = new GSRComponent();
   ```

   For more information, see Sending a request using the EMXMLInvoker or EMBasicXMLInvoker (on page 42).
5. Create an **EMXMLInvoker** instance and submit the following request.

   ```
   EMXMLInvoker my_invoker = new EMXMLInvoker(gsr_comp);
       try {
           XMLResponse = my_invoker.invoke(XMLRequest);
       }
   ```

```
    catch(InvokeException i)  {
    }
```

The class is now ready. You can use it in your project. For more information, see Sending a request using the EMXMLInvoker or EMBasicXMLInvoker (on page 42).

6. Compile your project.

   To run your project, you must pass it the relevant Java Virtual Machine variables.

   For more information, see Running your project (on page 17).

Sample class

The following example describes a sample class:

```
import com.bmc.ctmem.emapi.*;
public class EMAPISample {
  public EMAPISample() {
  }
  /** run once before submitting requests */
  public void do_init(String[] args)  {
      EMXMLInvoker.init(args);
  }
  /** run once before exiting the program */
  public void do_terminate() {
      EMXMLInvoker.done();
  }
  /** This submits the XMLRequest received as a parameter
   *  and returns the response  */
  public String submit_request(String XMLRequest)  {
  String  XMLResponse="";
  // Creates a   component
  ComponentType gsr_comp = new GSRComponent();
  // Creates a new   EMXMLInvoker instance
  EMXMLInvoker my_invoker = new EMXMLInvoker(gsr_comp);
  try {
    // Submits the request given as a parameter
    XMLResponse = my_invoker.invoke(XMLRequest);
  }
```

```
  catch(InvokeException i)  {
  // must handle InvokeException
  }
  return XMLResponse;
  }
}
```

Running your project

This procedure describes how to run your project.

➢ To run your project:

1. Set the environment variables for the Control-M/EM API by running the **emapi_env.bat** (or **emapi_env.sh**) file.
   - For **emapi_env.sh** on UNIX, run the following command:

     ```
     . emapi_env.sh
     ```

     Running the command with the above syntax ensures that the variables are valid after the command is run.
2. Ensure that the environment variables are set for the Java environment and that the path includes the Java installation location. If they are not set, run the following command:
   - **Microsoft Windows:**

     ```
     set JAVA_HOME=java_installation_location
     set PATH=java_installation_location;%PATH%
     ```
   - **UNIX:**

     ```
     setenv JAVA_HOME java_installation_location
     setenv PATH java_installation_location:$PATH
     ```
3. Copy the **emapi-900\ctmemapi.properties** file to your project working directory.
4. Run the java command from the project working directory, using the following CORBA parameters:
   - **Microsoft Windows:**

     ```
     java.exe -Dorg.omg.CORBA.ORBClass=org.jacorb.orb.ORB
     -Dorg.omg.CORBA.ORBSingletonClass=org.jacorb.orb.ORBSingleton
     -classpath
     %CLASSPATH% projectMainClass
     ```
   - **UNIX:**

     ```
     java -Dorg.omg.CORBA.ORBClass=org.jacorb.orb.ORB
     ```

```
-Dorg.omg.CORBA.ORBSingletonClass=org.jacorb.orb.ORBSingleton
-classpath
```

```
$CLASSPATH projectMainClass
```

projectMainClass is your project main class.

NOTE: These parameters are needed by Control-M/EM API. Pass them to your project as the first and second runtime parameters. The command must be entered as a single line.

Optionally, you can pass these parameters by using one of the alternative methods described in Error in Java virtual machine parameters (on page 190).

4

Control-M/EM API upgrade

This topic describes the changes to Control-M/EM API version 9.0.00. To update Control-M/EM, see Control-M Workload Automation Upgrade.

The following sections describe how to update Control-M/EM API and any changes that are necessary to update your applications and XML requests from previous versions of Control-M/EM API.

Updating Control-M/EM API

This procedure describes how to update Control-M/EM API, which enables users of previous versions to work in version 9.0.00.

Before you begin

Ensure you complete the following:

- Installing Control-M/EM API (on page 11)
- Configuring Control-M/EM API (on page 13)

➢ To update the Control-M/EM API:

1. Set your environment using one of the provided scripts:
   - Microsoft Windows: **emapi_env.bat**
   - UNIX **emapi_env.sh**

   NOTE: If you do not use the provided script, ensure that the startup script and the CLASSPATH point to the new classes.
2. Update your applications and XML requests.
3. (optional) Remove the **emapi-8***xx* directory from the previous version.

Changes from version 8.0.00 to version 9.0.00

Code developed for Control-M/EM API 8.0.00 is compatible with version 9.0.00 with the following exceptions:

- Control-M/EM API version 9.0.00 includes the following new parameters:
  - New day

- capture
- do_force job

NOTE: To use the parameters, set the namespace declaration to http://www.bmc.com/ctmem/schema900.

For more information on the parameters and requests, see Request and response parameters (on page 46) and Job and SMART Folder XML parameters.

Changes from version 7.0.00 to version 8.0.00

Code developed for Control-M/EM API 7.0.00 is compatible with version 8.0.00 with the following exceptions:

- Control-M/EM version 8.0.00 introduces new terminology: "folder" and "SMART Folder," which replaces the terms "table" and "SMART table." The new terminology is reflected in requests/responses that are described in the XML schema for version 8.0.00 (EMAPI_800.xsd).

The following table lists the elements that have been renamed in version 8.0.00 to reflect the new terminology:

version 7.0.00	version 8.0.00
autoedit	variable
sysout	output
owner	run_as
author	created_by
group	sub_application
over_lib	override_path
user_daily	order_method
force_ok	set_to_ok

The following requests/responses are affected by these changes:

- request_def_upload_folder/response_def_upload_folder
- request_order_force/response_order_force
- request_def_delete_jobs/response_def_delete_jobs
- request_create_aj
- response_act_retrieve_jobs

NOTE: To use the parameters, set the namespace declaration to http://www.bmc.com/ctmem/schema800.

For more information on the parameters and requests, see Request and response parameters (on page 46) and Job and SMART Folder XML parameters.

Changes from version 6.4.01 to version 7.0.00

Code developed for Control-M/EM API 6.4.01 is compatible with version 7.0.00 with the following exceptions:

- Control-M/EM 7.0.00 supports a nested hierarchy of tables. As a result, Control-M/EM API 7.0.00 introduced the following requests to support the nested table structure:
  - request_def_create_table
  - request_def_add_table
  - request_def_add_jobs

  The above requests replaced "request_def_create_jobs" and "request_def_create_sched_group" that were provided by Control-M/EM API for versions 6.3.00 and 6.4.00. The XML schema for version 7.0.00 does not support "request_def_create_jobs" and "request_def_create_sched_group."
- Control-M/EM version 7.0.00 introduced new terminology: "table" and "SMART Table," which replaced the terms "scheduling table" and "group scheduling table." The new terminology is reflected in requests/responses that are described in the XML schema for version 7.0.00 (EMAPI_700.xsd).

The following table lists the elements that have been renamed in version 7.0.00 to reflect the new terminology:

Earlier versions	Version 7.0.00
sched_table	table
scheduling_group_info	table_info
group_rba	table_rba
into_group	into_table
group_id	table_id
is_group	is_table

The following requests/responses are affected by these changes:

- request_def_upload_table/response_def_upload_table
- request_order_force/response_order_force
- request_def_delete_jobs/response_def_delete_jobs
- request_create_aj
- response_act_retrieve_jobs

NOTE: To use the parameters, set the `xmlns:ctmem namespace` declaration to http://www.bmc.com/ctmem/schema700.

Changes from version 6.3.0x to version 6.4.01

Code developed for Control-M/EM API version 6.3.0x is compatible with version 6.4.01 except for the following:

- The encodePassword method of EMXMLInvoker and EMBasicXMLInvoker is not supported in Control-M/EM API version 6.4.01. Replace all calls to the encodePassword method with calls to the BuildPasswordString method. For more information see BuildPasswordString (on page 27).
- Control-M/EM API version 6.4.01 includes the following new parameters that were added to Control-M/Server version 6.4.01.
  - cyclic_type
  - interval_sequence
  - specific_times
  - tolerance
  - attach_sysout

You can use these parameters when submitting the following requests:

- request_create_aj
- request_def_create_jobs
- request_create_sched_group

NOTE: To use the parameters set the `xmlns:ctmem namespace` declaration to http://www.bmc.com/ctmem/schema640.

Examples of an XML request using different formats

The following example is an XML request using the previous format:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
<ctmem:request_def_add_table
xmlns:ctmem="http://www.bmc.com/ctmem/schema700">
      <ctmem:user_token>$USER_TOKEN$</ctmem:user_token>
      <ctmem:parent_table>
            <ctmem:control_m>$CONTROL_M$</ctmem:control_m>
            <ctmem:table_name>smart</ctmem:table_name>
            <ctmem:table_library/>
            <ctmem:user_daily>user_daily</ctmem:user_daily>
      </ctmem:parent_table>
```

```
      <ctmem:sub_table>
            <ctmem:table_name>sub</ctmem:table_name>
            <ctmem:table_attributes>
                  <ctmem:application>apiWinApp3</ctmem:application>
                  <ctmem:group>apiGroup3</ctmem:group>
                  <ctmem:owner>controlm</ctmem:owner>
                  <ctmem:author>emuser</ctmem:author>
                  <ctmem:sub_table_rule_based_cals>
                        <ctmem:sub_rule_based_cal>
                              <ctmem:rule_based_cal_name>
                              rule_based_cal_name
                              </ctmem:rule_based_cal_name>
                        </ctmem:sub_rule_based_cal>
                  </ctmem:sub_table_rule_based_cals>
            </ctmem:table_attributes>
      </ctmem:sub_table>
</ctmem:request_def_add_table>
      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The following example is an XML request using the current format:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
<ctmem:request_def_add_folder
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
      <ctmem:user_token>$USER_TOKEN$</ctmem:user_token>
      <ctmem:parent_folder>
            <ctmem:control_m>$CONTROL_M$</ctmem:control_m>
            <ctmem:folder_name>smart</ctmem:folder_name>
            <ctmem:folder_library/>
            <ctmem:order_method>user_daily</ctmem:order_method>
      </ctmem:parent_folder>
      <ctmem:sub_folder>
            <ctmem:folder_name>sub</ctmem:folder_name>
```

```
            <ctmem:folder_attributes>
                  <ctmem:application>apiWinApp3</ctmem:application>
                  <ctmem:sub_application>apiGroup3
                  </ctmem:sub_application>
                  <ctmem:run_as>controlm</ctmem:run_as>
                  <ctmem:created_by>emuser</ctmem:created_by>
                   <ctmem:sub_folder_rule_based_cals>
                         <ctmem:sub_rule_based_cal>
                               <ctmem:rule_based_cal_name>
                               rule_based_cal_name
                               </ctmem:rule_based_cal_name>
                        </ctmem:sub_rule_based_cal>
          </ctmem:sub_folder_rule_based_cals>
        </ctmem:folder_attributes>
      </ctmem:sub_folder>
    </ctmem:request_def_add_folder>
      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Control-M/EM API Class references

This section describes class reference information about the functions that you can access when using Control-M/EM API in your project.

The following table describes the Control-M/EM API classes:

Class	Description
ComponentType (on page 25)	Defines an abstract base class, which represents a Control-M/EM component used with Control-M/EM API
EMXMLInvoker (on page 26)	Defines the primary class of the API
EMBasicXMLInvoker (on page 26)	Inherits from the EMXMLInvoker class
GSRComponent (on page 32)	Represents the Control-M/EM GUI Server
InvokeException (on page 33)	Enables you to process error feedback from Control-M/EM

ComponentType class

An abstract base class, which represents a Control-M/EM component used with Control-M/EM API.

When you send a request, the Control-M/EM component (which the request is refers to) is not indicated in the request. Instead, the ComponentType instance refers the request instance to the appropriate component.

The GSRComponent class (on page 32) is derived from the ComponentType class. This class is used when creating an instance of the **EMXMLInvoker** or the **EMBasicXMLInvoker**. For more information about the EMXMLInvoker and EMBasicXMLInkover see the following:

- EMXMLInvoker class (on page 40)
- EMBasicXMLInvoker class (on page 38).

EMBasicXMLInvoker and EMXML Invoker class

The **EMXMLInvoker** class is the primary class of the API. Its methods are used to initiate and stop the API, get and set API properties, and send requests to Control-M/EM. The **EMBasicXMLInvoke**r class is inherited from the **EMXMLInvoker** class and shares many of the methods of that class. For more information, see Control-M/EM API requests (on page 35).

The **EMBasicXMLInvoker** and the **EMXMLInvoker** classes use the following methods:

Method	Description
BuildPasswordString (on page 27)	Encodes a given text string for use in a Registration request.
done (on page 27)	Stops Control-M/EM API services by breaking the connection with the CORBA processes.
encodePassword	Use the **BuildPasswordString** (on page 27) method instead of the **encodePassword** method to encode a given text string for use in a Registration request. The **encodePassword** method is no longer supported in Control-M/EM API version 6.4.01 and above.
getProperties (on page 27)	Obtains for example, the Control-M/EM GUI Server names from the **ctmem.properties** file.
setProperties (on page 28)	Specifies, for example, the Control-M/EM GUI Server names and a location from which to obtain them.
Init (on page 29)	Starts the Control-M/EM API services and initializes CORBA with default values or with values specified with its optional parameters.
invoke (on page 31)	Enables you to send a request to Control-M/EM as a XML text string. When the invoke method is used with the **EMBasicXMLInvoker** class, polling for responses from Control-M/EM is automatic. When the invoke method is used with the **EMXMLInvoker** class, the user must activate polling to receive a response from Control-M/EM
setPollRequestIntervalMilli (on page 32)	(**EMBasic XMLInvoker** only) Determines the interval, in milliseconds, between automatic poll requests.
setPollRequestTimeoutMilli (on page 32)	(**EMBasic XLMInvoker** only) Determines the total time, in milliseconds, allotted for polling following a request.

BuildPasswordString

Enables you to prepare a text string for use as a user password in the User Registration request.

The **BuildPasswordString()** function must be called immediately before sending every API Register request because a different string is used for each request in some modes.

- Syntax: `Public String BuildPasswordString(password_string) throws InvokeException`
- Parameters: `password_string.` Text string, subject to all limitations that apply to a Control-M/EM password
- Return codes: The prepared string to be used in the password field of the registration request.
- See User Registration (on page 49)

done

Enables you to stop the Control-M/EM API services by breaking the connection with the CORBA processes (**EMXMLInvoker** static method).

- Syntax: `Public static void done()`
- Parameters: None
- Return codes: None
- See Configuring Control-M/EM API (on page 13) and Init (on page 29).

getProperties

Enables you to obtain the Control-M/EM GUI Server name from the **ctmem.properties** file. This method returns the properties object that you specify with the setProperties (on page 28) method. If you have not set the Control-M/EM API properties by using **setProperties**, it returns the properties object containing the values in the **ctmem.properties** file under your *project/application* working directory.

Each call to **getProperties** reads the **ctmem.properties** file. Any modifications made to this file while the application is running affect subsequent calls to **getProperties**.

- Syntax: `Public Properties getProperties();`
- Parameters: The following describes the parameters for **getproperties**:
  - **host_of_the_GUI_Server:** Name of the Control-M/EM GUI Server. Default: **com.bmc.ctmem.emapi.GSR.hostname**
  - **path_to_XML_data_files:** Location of the XML schema. Default: **com.bmc.ctmem.emapi.XMLDATAPAT**
- Return codes: properties

setProperties

Used to specify the Control-M/EM Global Alerts Server or GUI Server host names and the source from which to obtain them.

- Syntax: `Public void setProperties(Properties props);`
- Parameters: The following table describes the parameters for SetProperties:

Parameter	Description
host_of_the_GUI_Server	Set to the hostname of the Control-M/EM GUI Server. Default: **com.bmc.ctmem.emapi.GSR.hostname**
host_of_the_Alerts_Server	Set to the hostname of the Global Alerts Server. Default: **com.bmc.ctmem.emapi.GAS.hostname**
path_to_XML_data_files	Location of the XML schema. Default: **com.bmc.ctmem.emapi.XMLDATAPATH**

- Return codes: None
- See the following:
  - getProperties (on page 27)
  - Getting and setting Control-M/EM API properties

Init

Starts the Control-M/EM API services and initializes CORBA with default values or with values specified with its optional parameters (an **EMXMLInvoker** static method). The `init` method must be run at program startup. The following table describes the three different prototypes.

init	Description
Prototype 1	Default implementation which initializes the CORBA services using a Control-M/EM API-specific CORBA configuration that was created during the API post-installation configuration (using **emapi-configure**). ▪ Syntax: `Public static void init()` ▪ Parameters: None ▪ Return codes: None This CORBA configuration information is contained in the **jacorb.properties** file located under the **emapi-900\etc** directory.
Prototype 2	Enables you to include an array of strings representing a list of arguments. ▪ Syntax: `Public static void init(String[] args)` ▪ Parameters: The **args** parameter generally contains your command line arguments for the application's `main`. This enables you to control ORB initialization from outside the program. For a list of CORBA parameters suitable for use in `args`, see the manufacturer's documentation. ▪ Return codes: None ▪ For Code and Run examples, see Example: Init Prototype 2 (on page 30).

init	Description
Prototype 3	Enables you to include an array of strings representing a list of arguments. ▪ Syntax: `Public static void init(String[] args, Properties props)` ▪ Parameters: The **args** parameter generally contains your command line arguments for the application's `main`. This enables you to control ORB initialization from outside the program. The `props` parameter (Properties) can contain CORBA parameters, using the same options as in the command line that was passed as the first parameter (`args`). For a list of CORBA parameters suitable for use in `args`, see the manufacturer's documentation. ▪ The Properties class is part of the **java.util** package. ▪ Return codes: None ▪ For Code and Run examples, see Example: Init Protoype 3 (on page 30).

Example: Init Prototype 2

The following is a code example:

```
public class HelloWorld {
    public static void main(String[] args) {
        EMXMLInvoker.init(args);
        ...
    }
}
```

The following is a run example:

```
java HelloWorld -jacorb.implname StandardNS
```

Example: Init Protoype 3

The following is a code example:

```
public class HelloWorld {
    public static void main(String[] args) {
Properties props = new Properties();
props.setProperty("jacorb.implname", "StandardNS");
```

```
        EMXMLInvoker.init(args, props);
        ...
    }
}
```

The following is a run example:

```
java HelloWorld
```

NOTE: This run example is used for illustrative purposes. To run it, you must add Java virtual machine (JVM) parameters and the Control-M/EM API CLASSPATH. These concepts are discussed in Control-M/EM API Installation (on page 10). This CORBA configuration information is contained in the **jacorb.properties** file located in the **emapi-900\etc** directory.

By using both values for props and args, you can supply values for props that can be overridden by values that are supplied for args.

invoke

Enables you to send a request to Control-M/EM. The request is sent as an XML text string. The invoke method is used with either the **EMBasicXMLInvoker** class or **EMXMLInvoker** class. A request in XML format (`xml request`), specifying the action that Control-M installation is to perform, is required for each call.

- Syntax: `public String invoke(String xmlRequest) throws InvokeException`
- Parameters: The **xmlrequest** string is a request that you send to Control-M/EM. The string is a text file in an XML format that the Control-M/EM API can accept and interpret.
- Return codes: Response in XML format. Response data that addresses the request that was sent. It is returned as an XML formatted string.

  See invoke and Request and response parameters (on page 46)

Polling interval timeout configuration

Requests that are sent using the **EMBasicXMLInvoker** class are polled automatically with default polling values. Poll requests are sent every five seconds until a reply is received. It is recommended that you change the polling values to your network capabilities. You can modify the number and frequency of poll requests that are sent using the **setPollRequestIntervalMilli** (on page 32) and **setPollRequestTimeoutMilli** (on page 32) methods.

The total number of times that polling is conducted is a function of the values determined by the **setPollRequestIntervalMilli** and **setPollRequestTimeoutMilli** methods and the amount of time that each poll request takes.

EXAMPLE: The total amount of time for polling is set at 10,000 milliseconds (10 seconds), using the **setPollRequestTimeoutMilli** method.

The time between poll requests is set at 2000 milliseconds (2 seconds), using the **setPollRequestintervalMilli** method.

Each polling request takes about 500 milliseconds (0.5 seconds).

10,000 / (2000 + 500) = 4 poll requests

setPollRequestIntervalMilli

(EMBasicXMLInvoker only) Sets the interval between automatically-sent poll requests. This time is measured from the **end** of the current poll request. This value must be less than or equal to the value specified with the setPollRequestTimeoutMilli (on page 32) method. See Polling interval timeout configuration (on page 31).

- Syntax: `Public void setPollRequestIntervalMilli (final long interval)`
- Parameters: **interval**. Time, in milliseconds.
- Default: **5000** milliseconds (5 seconds)
- Return codes: None

setPollRequestTimeoutMilli

(EMBasicXMLInvoker only) Sets the total time that is allotted for polling for a response to a request. This value must be greater than or equal to the value specified with the setPollRequestIntervalMilli (on page 32) method.

- Syntax: `Public void setPollRequestTimeoutMilli(final long timeout)`
- Parameters: **timeout**. Time, in milliseconds
- Default: -1 milliseconds (the -1 value indicates that polling is carried on until a response is received. There is no timeout value).
- Return Codes: None
- See Polling interval timeout configuration (on page 31).

GSRComponent class

Represents the GUI Server. When creating an instance of **EMXMLInvoker** (or **EMBasicXMLInvoker**) with a reference to a GSRComponent type, the request is sent to the specified Control-M/EM GUI Server. For more information, see ComponentType class (on page 25).

The GSRComponent class uses the constructors listed in the following table:

Constructor	Description
GSRComponent (prototype 1)	Using this constructor, the request is sent to the GUI Server name specified under **com.bmc.ctmem.emapi.GSR.name=***name* in the **ctmemapi.properties** file. ▪ Syntax: `GSRComponent();` ▪ Parameters: None ▪ Return codes: None
GSRComponent (prototype 2)	Using this constructor, the request is sent to the GUI Server name specified with the **name** parameter. ▪ Syntax: `GSRComponent(String name);` ▪ Parameters: **name**. This is the GUI Server name. ▪ Return codes: None

InvokeException class

Enables Control-M/EM API user to obtain error information when an exception is thrown.

For error codes and exceptions see Error codes and exceptions (on page 195).

The **InvokeException** class includes the methods listed in the following table:

Method	Description
getMajorCode	Enables you to obtain the Major Code that identifies the error family to which an error belongs. ▪ Syntax: `Public int getMajorCode()` ▪ Parameters: None ▪ Return codes: `int`. An integer that identifies the error family to which the error belongs.
getMinorCode	Enables you to obtain the Minor Code of an error. The Minor Code provides a unique identifier for the error in the family to which it belongs. ▪ Syntax: `Public int getMinorCode()` ▪ Parameters: None ▪ Return codes: `int`. An integer that provides a unique identifier for the error in its family.
getReason	Enables you to obtain the text description of an error. ▪ Syntax: `Public String getReason()` ▪ Parameters: None ▪ Return codes: String. This string is a text description of the error.

6

Control-M/EM API requests

Control-M/EM API supports two types of requests:

- Synchronous Requests (on page 36): Synchronous requests are processed by Control-M/EM and responses are received directly from Control-M/EM.
- Asynchronous Requests (on page 37): Asynchronous requests are sent through Control-M/EM to Control-M/Server. By sending a polling request that includes the tracking ID, the user can retrieve feedback about what action Control-M/EM takes, or notice that the request is still being processed.

  Before sending requests, Control-M/EM API must be initialized. Control-M/EM supports two methods for creating, sending and handling requests. See Control-M/EM API programming methods (on page 37) for more information.

Synchronous Requests

Synchronous requests are processed by Control-M/EM. Responses are received directly from Control-M/EM. These include some of the following:

Request	Description
polling requests	Defines responses from Control-M/EM.
user registration request (on page 49)	Sends the username and password of the user to the target server component.
Check user token validity (on page 51)	Checks if the specified user identification is still valid.
User Unregistration (on page 55)	Enables an unregister request to send the user token to the server component when the application finishes using the API.
Retrieve jobs in the active jobs database (on page 162)	Enables you to retrieve jobs that are in the active jobs database.
Change alert status (on page 177)	Changes the status (for example, from **not_noticed** to **handled**) of an alert.
Job tracking (on page 156)	Tracks the progress of existing jobs in Control-M installation.
create job and SMART folder definitions (on page 57)	Creates a new outer-most folder (regular or SMART Folder) definitions.
Delete job definitions (on page 85)	Deletes one or more jobs from a folder according to the user defined deletion criteria. SMART Folder or sub-folder entities are not deleted.
Retrieve BIM Services list (on page 180)	Retrieves the list of services active in the Batch Impact Manager Server.
Request list conditions (on page 148)	Retrieves a list of conditions for a job.

Asynchronous Requests

Asynchronous requests are sent through Control-M/EM to Control-M/Server. These requests receive a tracking ID as a response. By sending a polling request that includes this tracking ID, the user can retrieve feedback about what action Control-M/EM takes, or a notice that the request is still being processed. The following table describes asynchronous requests:

Request	Description
Order or Force Request (on page 94)	Enables you to force an individual job or all jobs in a folder to be placed in Active Jobs.
add (on page 141) condition Delete condition (on page 144) request	Enables you to add or delete prerequisite conditions to jobs.
Job creation (on page 106)	Enables you to create a job or a SMART Folder and inserts the job into Active Jobs.
Upload folder (on page 90)	Enables you to upload an outermost folder.
Job actions in active jobs (on page 151)	Enables you to perform actions on jobs that are currently in the active jobs database.

Control-M/EM API programming methods

Control-M/EM API supports two methods for creating, sending, and handling requests:

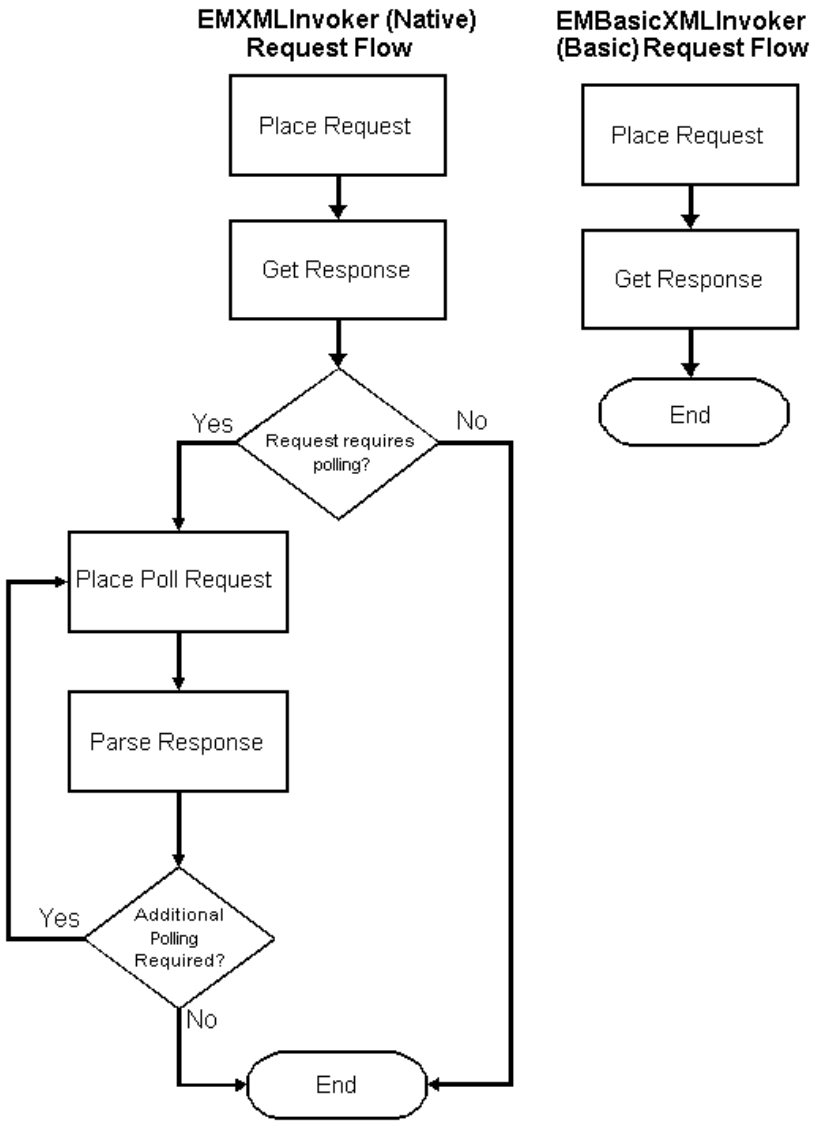
- EMXMLInvoker (on page 40): Enables Control-M/EM to send and receive XML requests using Control-M/EM API, which requires you to poll manually for responses. The `init, done, invoke, getProperties` and `setProperties` methods are used with this class.
- EMXMLBasicXMLInvoker (on page 38): Enables Control-M/EM to send and receive XML requests using Control-M/EM API, which polls automatically for responses (unlike EMXMLInvoker), and eliminates the effort of polling by not returning the response until either the final response is available from the server, or the API times out.

  The EMBasicXMLInvoker uses an XML schema to validate XML formatted requests and responses, but the EMXMLInvoker class does not (see XML string validation (on page 39)). Both classes can be used together in the same project, and in the same session. As the EMBasicXMLInvoker class inherits from the EMXMLInvoker class you only need to initialize the Control-M/EM API once. See Flowchart: Differences between EMXMLInvoker and EMBasicXMLInvoker (on page 38) for more information.

  To send a request using EMXMLInvoker or EMBasicXMLInvoker, see Sending a request using the EMXMLInvoker or EMBasicXMLInvoker (on page 42). For response types, see Response types received when using the EMBasicXMLInvoker (on page 44) and Response types received when using EMXMLInvoker (on page 45).

Flowchart: Differences between EMXMLInvoker and EMBasicXMLInvoker

The following figure illustrates the structural differences between the two request methods:



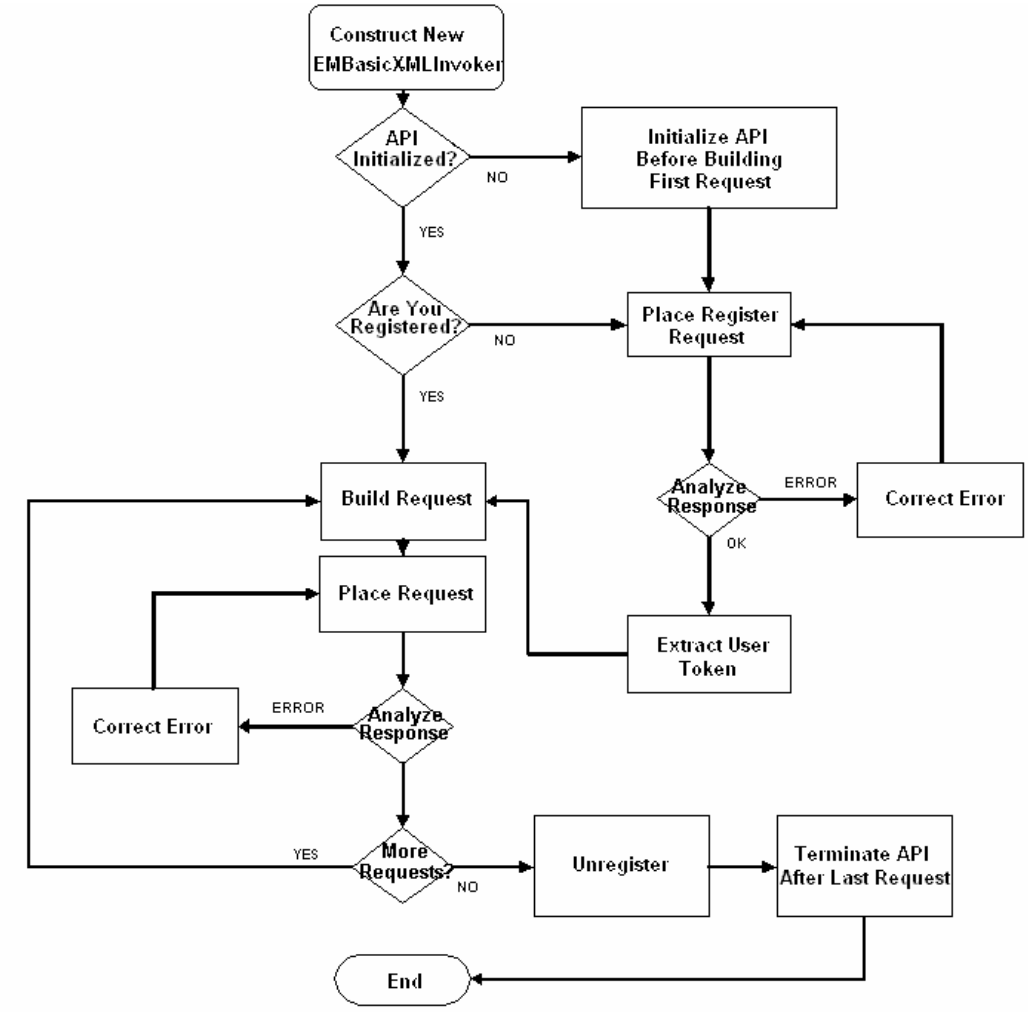
EMBasicXMLInvoker class

The EMBasicXMLInvoker class eliminates the need for polling by not returning the response until either the final response is available from the server, or the API times out. The advantage is that you do not have to process the responses or place additional requests. The EMBasicXMLInkvoker uses an XML schema to validate XML formatted request and response strings. See XML string validation (on page 39).

The EMBasicXMLInvoker class has the following limitations:

- The API call might be blocked for several seconds, until the final response is available or the call times out. The time out period is configurable.
- After receiving the response, the EMBasicXMLInvoker class performs additional processing to determine its course of action.
- Because the EMBasicXMLInvoker class performs XML parsing, this class makes it more resource intensive than the EMXMLInvoker class.

The following flowchart illustrates the EMBasicXMLInvoker request session:



XML string validation

The EMBasicXMLInvoker uses an XML schema to validate XML formatted request and response strings. The location of this schema is determined by the **com.bmc.ctmem.emapi.XMLDATAPATH** property, which is defined in the **ctmemapi.properties file**. If this path is not available to the class at run time, the operation fails.

The EMXMLInvoker class does not examine the XML formatted data that it sends or receives, and it does not depend on the presence of the XML schema.

The XML standard does not include support for the following characters. If these characters are used, they are translated in the XML file as listed in the following table:

Character	Meaning	Translated to
<	less than	<
>	greater than	>
&	ampersand	&
"	quotation marks	"
‘	apostrophe	'
ASCII 10	line feed
ASCII 13	carriage return

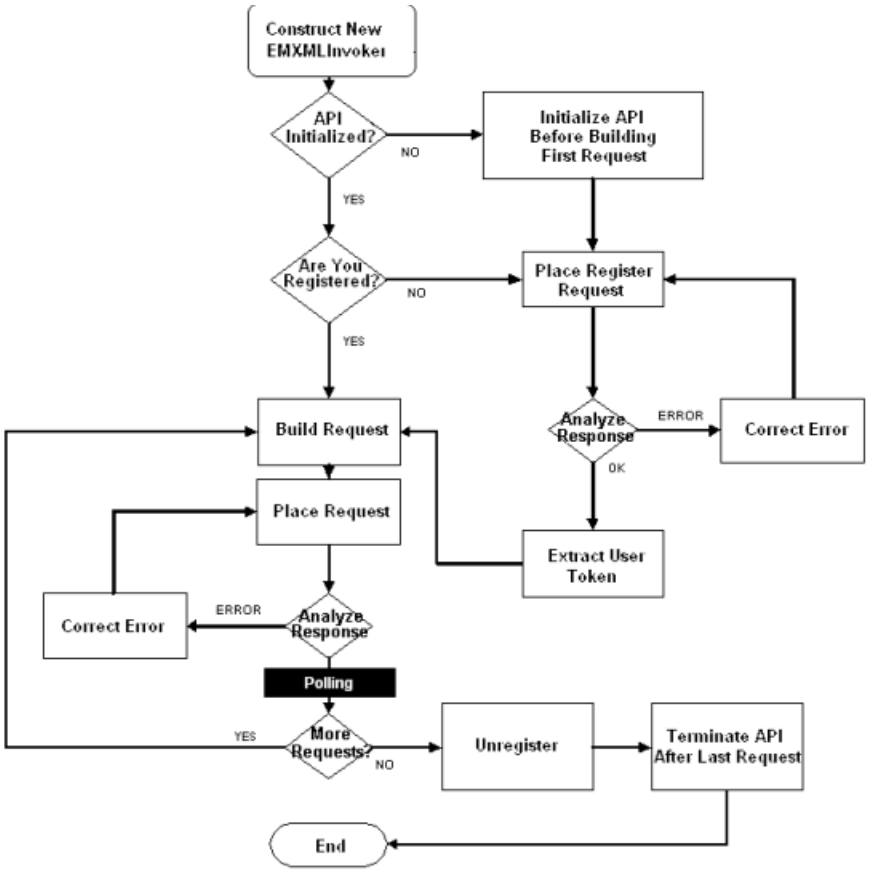
For information about using the EMXMLInvoker and EMBasicXMLInvoker see Control-M/EM API requests (on page 35).

EMXMLInvoker class

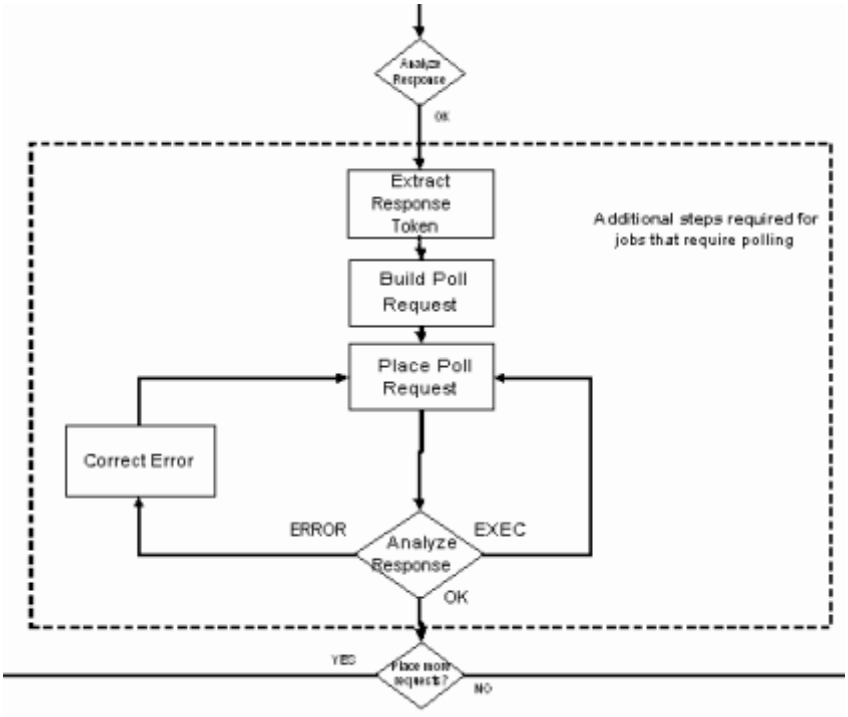
The EMXMLInvoker class sends XML formatted requests to Control-M/EM and returns responses to the program that issues requests. This class does not process response data. Certain asynchronous requests (such as the Job Creation request) take time for Control-M to process and only indicates if the request has been submitted successfully.

An application using the EMXMLInvoker class may make several requests to check whether the expected response is available by sending a poll request. To identify the original request (for which polling is being performed) the poll request includes a tracking ID, which is supplied in a response from Control-M/EM immediately after certain asynchronized requests are submitted. The program can submit this tracking ID in a poll request several times, until the required response is available.

The following flowchart illustrates an EMXMLInvoker request session:



The following figure illustrates the polling process in detail. The advantage of the polling process is that the API always replies immediately to the request. No extra time or resources are spent processing the responses.



Sending a request using the EMXMLInvoker or EMBasicXMLInvoker

This procedure describes how to send a request using the `EMXMLInvoker or EMBasicXMLInvoker`. The invoke (on page 33) method can throw an exception if the application fails to process the invoke call (for example, if communication between Control-M/EM and Control-M fails). For more information, see Application runtime and communication troubleshooting (on page 191).

➢ To send a request:

1. Create an instance of either the `EMXMLInvoker or EMBasicXMLInvoker` containing a reference to a Control-M/EM GUI server.

- **EXAMPLE:** EMXMLInvoker:

```
GSRComponent gsrComponent = new GSRComponent();
EMXMLInvoker gsrInvoker = new EMXMLInvoker(gsrComponent);
```

- **EXAMPLE:** EMBasicXMLInvoker:

```
GSRComponent gsrComponent = new GSRComponent();
EMBasicXMLInvoker gsrInvoker = new
EMBasicXMLInvoker(gsrComponent);
```

NOTE: In networks in which more than one GUI Server is installed, only the component is listed in the **ctmemapi.properties** file. The API works only with the components listed in the file. You cannot modify the file to include more than one GUI Server.

2. Send a request to the specified Control-M/EM server component according to the different class:

- **EXAMPLE:** EMXMLInvoke or EMBasicXMLInvokerr:

```
String xmlRequest = "<?xml?>..."; // xml request
String xmlResponse;
try{
    xmlResponse = gsrInvoker.invoke(xmlRequest);
}
catch(InvokeException ex){
    // handle invoke failures
}
// handle xml response
```

NOTE: A request in XML format (xmlRequest), specifying the action that the Control M installation is to perform, is required for each call. The various types of requests that you can make are described in Request and response parameters (on page 46).

Response types received when using the EMBasicXMLInvoker

The following table shows the types of responses you receive when using the EMBasicXMLInvoker:

Response type	Description
response_*operationName* **EXAMPLE:** response_unregister	This response is the final result of a successful synchronous request. The first field is **status** and the value is **OK** except for response_track_jobs where it can also be **PARTIAL_SUCCESS**. For more information see the note in SOAP envelope for Control-M/EM requests and responses (on page 46).
fault_*operationName* **EXAMPLE:** fault_unregister	Synchronous requests: This response indicates a failure. Asynchronous requests: This response indicates a failure in sending the request to the Control-M/Server. For example, if the user is not authorized, Control-M/Server is unavailable, does not have the right status, or the request has invalid values.
response_poll_*operationName* **EXAMPLE:** response_poll_add_condition	This response is the final successful or partially successful synchronous request. The first field is **status** and the value is **OK** except for response_poll_order_force where it can also be **PARTIAL_SUCCESS**. For more information see the note in SOAP envelope for Control-M/EM requests and responses (on page 46).
fault_poll_*operationName* **EXAMPLE:** fault_poll_add_condition	This response is the result of an error on an asynchronous request that was sent to the Control-M/Server. This response can be an error returned from the Control-M/Server itself (for example, an order request for a nonexistent table), or an error from the communication layer between Control-M/EM and Control-M/Server (for example, a request timeout or communication loss).

Response types received when using EMXMLInvoker

The following table shows the types of responses you receive when using the EMXMLInvoker:

Response type	Description
response_*operationName* **EXAMPLE:** response_unregister	This response is the final result of a successful synchronous request. The first field is **status** and the value is **OK** except for response_track_jobs where it can also be **PARTIAL_SUCCESS**. For more information see the note in SOAP envelope for Control-M/EM requests and responses (on page 46). Asynchronous requests: The response contains the token ID that should be used to poll Control-M/EM for the final result of the request.
fault_*operationName* **EXAMPLE:** fault_unregister	Synchronous requests: This response indicates a failure. Asynchronous requests: This response indicates a failure in sending the request to the Control-M/Server. For example, if the user is not authorized, Control-M/Server is unavailable, does not have the right status, or the request has invalid values.
response_poll_*operationName* **EXAMPLE:** response_poll_add_condition	This response is received as an answer to a polling request. The first field is **status**. A value of EXEC indicates that the request is still being processed by the Control-M/Server and the response has not yet arrived. A value of either **OK** or **PARTIAL_SUCCESS** indicates the final response of an asynchronous request. The value **PARTIAL_SUCCESS** can appear only in **response_poll_order_force**. For more information, see the note in SOAP envelope for Control-M/EM requests and responses (on page 46).
fault_poll_*operationName* **EXAMPLE:** fault_poll_add_condition	This is an error response, which is received from a polling request. This response can be an error returned from Control-M/Server. For example, an order request for a nonexistent table, or an error from the communication layer between Control-M/EM and Control-M/Server, such as a request timeout or communication loss.

Request and response parameters

This topic describes Control-M/EM API request and response parameters. Many XML elements in this section are Control-M job parameters. To create a successful request, such as a Job Creation request or an Order Job request, you should become familiar with Control-M job parameters. See General parameters.

For Control-M/EM API request parameters, see Request XML parameters (on page 48).

NOTE: Control-M/EM API does not support non-English characters in request and response parameters.

Every time a request is made, Control-M/EM replies by sending one or more response strings. You can parse the contents of these response strings for use in your project. Control-M/EM API responses are wrapped by the SOAP envelope. For more information, see SOAP envelope for Control-M/EM requests and responses (on page 46).

SOAP envelope for Control-M/EM requests and responses

SOAP ("Simple Object Access Protocol") is a simple XML-based protocol that allows applications to exchange information. For more information, see www.w3.org.

Control-M/EM API uses SOAP envelopes to wrap the Control-M/EM requests and responses. Successful responses appear directly below the SOAP-ENV:Body node and contain the information requested or indicate success. Every request has a specific ctmem response. For example, ctmem:response_unregister is the response for a ctmem:request_unregister request.

FAULT responses are wrapped within a SOAPFAULT element and report failed requests. The fault node is located below the SOAP detail node. The fault node contains information about the errors that caused the operation to fail. Every request has a specific fault response. For example, ctmem:fault_unregister is the fault response for the ctmem:request_unregister request.

The following topics describe successful request and response examples together with a fault response:

- SOAP request example (on page 47)
- SOAP response example (on page 47)
- SOAP fault response example (on page 47)

NOTE: When Control-M/EM API runs a request that applies to multiple jobs, the request may not be successful on all jobs. The general status field returns a successful response and receives a value of **PARTIAL_SUCCESS** instead of **OK.** Each individual job status is either **OK** or **ERROR**. A **PARTIAL_SUCCESS** can only occur when the Control-M/EM API runs the following requests:

- track job
- order or force job

SOAP request example

The following example describes a SOAP request:

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
. . . SPECIFIED_REQUEST . . .
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP response example

The following example describes a SOAP response:

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
. . .SPECIFIED_REQUEST_RESPONSE . . .
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP fault response example

The following example describes a SOAP fault response:

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
<SOAP-ENV:Fault>
<faultcode> … </faultcode>
<faultstring>…. </faultstring>
<detail>
. . . FAULT_RESPONSE_OF_THE_REQUEST . . .
</detail>
</SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Request XML parameters

Requests are sent using the invoke (on page 31) method of EMXMLInvoker or EMBasicXMLInvoker. For practical examples, see Control-M/EM API flowchart (on page 7)

The following table lists the describes Control-M/EM API XML requests:

Name	Description
User Registration (on page 49)	Sends the username and password of the user to the target server component.
Check user token validity (on page 51)	Checks to see if the specified user identification is valid.
Client Keep Alive (on page 53)	Resets the registration timeout counter to zero.
User Unregistration (on page 55)	Enables you to see if the specified user identification is valid.
Create folder definitions (on page 57)	Creates a regular or SMART Folder definitions.
Add jobs to folder definitions (on page 78)	Enables you to add job definitions into an existing outermost folder or sub-folder.
Add folder to folder definitions (on page 82)	Enables you to add sub-folder definitions into an existing outermost SMART folder or sub-folder.
Delete job definitions (on page 85)	Deletes job definitions from the specified outermost or sub-folder.
Upload folder (on page 90)	Enables you to upload a folder
Order (on page 94)	Inserts a job or folder into Active Jobs immediately (force) or subject to scheduling criteria (order).
Job creation (on page 106)	Creates a job, SMART Folder, or sub-folder into the Active Jobs database.
Add condition (on page 141)	Enables you to add conditions for a job
Delete condition (on page 144)	Enables you to delete conditions for a job.
List conditions (on page 148)	Retrieves a list of conditions for a job.

Name	Description
Job actions in active jobs (on page 151)	Performs actions on jobs that are currently in the active jobs database.
Job tracking (on page 156)	Enables you to Poll the Response repository in the Control-M/EM GUI Server to receive completion confirmation from earlier job processing requests.
Retrieve jobs in the active jobs database (on page 162)	Enables you to retrieves jobs that are currently in the active jobs database.
Change alert status (on page 177)	Enables you to change the status of an alert (for example, from **not_noticed** to **handled**).
Retrieve BIM Services list (on page 180)	Retrieves the list of services active in the BMC Batch Impact Manager server.

User Registration

Sends the username and password of the user to the target server component. The server component returns a user token, which must accompany all subsequent requests made during the session.

Each Control-M/EM API request must contain a user token. This token is obtained using a register request, containing Control-M/EM user name and password. Upon successful authentication a new session is created for this user. This session is identified by the returned user token. When the session finishes, send an unregister request to log out and release resources allocated for this user session.

NOTE: A program can use multiple user tokens and issue several requests at once (for example, for a multi-threaded program). However, you should not issue another request with the same user token if the previous request is still in session.

The following topics describe the request user registration parameters, the response from Control-M/EM, together with examples:

- request_register XML parameters (on page 50)
- request_register XML example (on page 50)
- response_register parameters (on page 51)
- response_register XML example (on page 51)

request_register XML parameters

The following table describes the request_register XML parameters:

Parameter	Description
component	For Control-M for Web Services, Java, and Messaging only. See the relevant product documentation for more information.
user_name	Defines the Control-M/EM username of the person making the request. String.
password	Defines the Control-M/EM user password of the person making the request. **NOTE:** This password must be sent as an encrypted string. Therefore, you must use the BuildPasswordString method to encrypt the password prior to making the User Registration request. For more information see, BuildPasswordString (on page 27).
timeout	Indicates a length of time, in seconds, until the user's current user token is automatically invalidated. Integer. Default: 720. Optional. You should synchronize the value of this parameter with that of the EM_REFRESH_INTERVAL environment variable in Control-M/Enterprise Manager. For more information, refer to Preparing the Control-M/EM API project environment (on page 15). **NOTE:** Use the Timeout Reset Request to restart the count from 0.

For an example of the Request, see request_register XML example (on page 50).

request_register XML example

The following example describes a successful request:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

    <SOAP-ENV:Body>

        <ctmem:request_register
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

            <ctmem:user_name><user to register></ctmem:user_name>

            <ctmem:password><encrypted password></ctmem:password>

            <ctmem:timeout>40</ctmem:timeout>

        </ctmem:request_register>

    </SOAP-ENV:Body>
```

```
</SOAP-ENV:Envelope>
```

response_register parameters

The following table describes the response_register XML parameters:

Parameter	Description
status	Indicates a description of message content (for example, Error). String.
user_token	Indicates a unique ID that identifies the user. String.
authentication_message	Refers to a free text message containing authentication information. String

For an example of the response, see response_register XML example (on page 51). For error codes: See Authorization request errors (Major code 407) (on page 206).

NOTE: XML parameters for fault_register, and a sample fault response are described in Fault Response (on page 185).

response_register XML example

The following example describes a successful response:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <ctmem:response_register
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
            <ctmem:status>OK</ctmem:status>
            <ctmem:user_token>12345630</ctmem:user_token>
            <ctmem:authentication_message/>
        </ctmem:response_register>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Check user token validity

Enables you to check if the specified user identification is still valid.

The following topics describe the request check user token parameters, the response from Control-M/EM, together with examples:

request_check_user_token XML parameters

The following table describes the request_check_user_token XML parameters:

Parameter	Description
component	For Control-M for Web Services, Java, and Messaging only. See the relevant product documentation for more information.
user_token	Defines the serial identification number supplied to the user during registration. String.

For an example of a request, see request_check_user_token XML example (on page 52).

request_check_user_token XML example

The following example describes a successful request:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

    <SOAP-ENV:Body>

        <ctmem:request_check_user_token
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

            <ctmem:user_token>12345630</ctmem:user_token>

            </ctmem:request_check_user_token>

        </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

response_check_user_token XML parameters

The following table describes the table response_check_user_token XML parameters:

Parameter	Description
status	Indicates the status token validity. String. Valid values: ▪ **VALID** ▪ **INVALID**
will_expire	Indicates the amount of time, in seconds, remaining until the expiration of the user identification token. If the status is **INVALID** this parameter is set to zero.

For an example of a response, see response_check_user_token XML example (on page 53). For error codes: See Authorization request errors (Major code 407) (on page 206).

NOTE: The XML parameters for fault_check_user_token, and a sample fault response are described in Fault Response (on page 185).

response_check_user_token XML example

The following example describes a successful response:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <ctmem:response_check_user_token
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
            <ctmem:status>OK</ctmem:status>
            <ctmem:will_expire>587</ctmem:will_expire>
        </ctmem:response_check_user_token>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Client Keep Alive

When a Control-M/EM user registers with Control-M/EM API, the user receives a user token that is in effect for a limited period of time. The Timeout Reset request resets the registration timeout counter to zero. Timeout Reset requests can be sent intermittently to keep a user's registration valid during lengthy sessions.

The following topics describe the request client keep alive parameters, the response from Control-M/EM, together with examples:

request_client_keep_alive XML parameters

The following table describes the request_client_keep_alive XML parameters:

Parameter	Description
component	For Control-M for Web Services, Java, and Messaging only. See the relevant product documentation for more information.
user_token	Defines the serial identification number supplied to the user during registration. The user indicated by this number is the user. String.

For an example of a request, see request_client_alive XML example (on page 54).

request_client_alive XML example

The following example describes a successful request:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

    <SOAP-ENV:Body>

        <ctmem:request_client_keep_alive
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

            <ctmem:user_token>12345630</ctmem:user_token>

        </ctmem:request_client_keep_alive>

    </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

response_client_keep_alive XML parameters

The following table describes the response_client_keep_alive XML parameters:

Parameter	Description
status	Indicates the condition of the element that is contained within it (for example, Error). String.
user_token	Indicates a Unique ID that identifies the user. String.

For an example of a response, see response_client_alive XML example (on page 55). For error codes: See Authorization request errors (Major code 407) (on page 206).

NOTE: XML parameters for fault_client_keep_alive, and a sample fault response are described in Fault Response (on page 185).

response_client_alive XML example

The following describes a successful response:

```
<?xml version="1.0" encoding="iso-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

    <SOAP-ENV:Body>

        <ctmem:response_client_keep_alive
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

            <ctmem:status>OK</ctmem:status>

        </ctmem:response_client_keep_alive>

    </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

User Unregistration

When the application finishes using the API, you send an Unregister request, which in turn, sends the user token to the server component. The server component erases the user from its active users list. The user token is invalidated when the request is complete.

The following topics describe the request user unregistration parameters, the response from Control-M/EM, together with examples:

- request_unregister XML parameters (on page 56)
- request_unregister XML example (on page 56)
- response_unregister XML parameter (on page 56)
- response_unregister XML example (on page 57)

request_unregister XML parameters

The following table describes the request_unregister XML parameters:

Parameter	Description
component	For Control-M for Web Services, Java, and Messaging only. See the relevant product documentation for more information.
user_token	Defines a unique ID that identifies the user. String.

For an example of the request, see request_unregister XML example (on page 56).

request_unregister XML example

The following describes a successful request:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

    <SOAP-ENV:Body>

        <ctmem:request_unregister
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

            <ctmem:user_token>12345630</ctmem:user_token>

        </ctmem:request_unregister>

    </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

response_unregister XML parameter

The following table describes the response_unregister XML parameters:

Parameter	Description
status	Indicates a description of message content (for example, Error). String.

For an example a response, see response_unregister XML example (on page 57). For error codes, see Authorization request errors (Major code 407) (on page 206).

NOTE: XML parameters for fault_unregister, and a sample fault response are described in Fault Response (on page 185).

response_unregister XML example

The following example describes a successful response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

  <SOAP-ENV:Body>

    <ctmem:response_unregister
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

      <ctmem:status>OK

      </ctmem:status>

    </ctmem:response_unregister>

  </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Create folder definitions

Enables you to create new outer-most folder (regular or SMART Folder) definitions. It is possible to create only one folder per request. If the specified folder already exists, the request fails. It is possible to create an empty folder or a folder containing jobs or sub-folders (in the case of SMART Folders).

The following topics describe the request to create folder definitions parameters, the response from Control-M/EM, together with examples:

- request_def_create_folder XML parameters (on page 58)
- response_def_create_folder XML parameters (on page 72)
- Create folder definitions examples (on page 72)

request_def_create_folder XML parameters

The following table describes the request_def_create_folder XML Parameters:

Parameter	Description
user_token	Defines a serial identification number supplied to the user during registration. String.
folder	Defines a folder wrapper, which is identified by the elements in folder_type XML parameters (on page 58).
SMART_folder	Defines a SMART Folder wrapper. The folder is identified by the elements in sub_folder_type XML parameters (on page 59).

For examples of requests, see Create folder definitions examples (on page 72).

folder_type XML parameters

The following table describes the folder XML parameters:

Parameter	Description
control_m	Defines the name of the Control-M installation that processes the request. String.
folder_name	Defines the name of the outermost folder. String.
folder_library	(Control-M for z/OS only) Defines the name of the library in which the table is located. String
order_method	Defines the name of the user daily. String.
jobs	Defines a sequence of job elements. See Job and SMART Folder XML parameters (on page 60) for a list of the parameters that can be used or specified in job elements.

SMART_folder_type XML parameters

The following table describes the SMART_folder XML parameters:

Parameter	Description
control_m	Name of the Control-M installation that processes the request. String.
folder_name	Name of the folder. String.
folder_library	**Control-M for z/OS only**: Name of the library in which the table is located. String
user_daily	Name of the user daily. String.
folder_attributes	The SMART Folder's attributes. See Job and SMART Folder XML parameters.
sub_folders	A sequence of sub-folder elements. See Job and SMART Folder XML parameters for a list of the parameters that can be used/specified in sub-folder elements.
jobs	A sequence of job elements. See Job and SMART Folder XML parameters for a list of the parameters that can be used/specified in job elements.

sub_folder_type XML parameters

The following table describes the sub-folder XML parameters:

Parameter	Description
folder_name	Defines the name of the sub-folder. String.
folder_attributes	Defines the sub-folder's user defined attributes. See Job and SMART Folder XML parameters (on page 60).
sub_folders	Defines a sequence of sub-folder elements. See Job and SMART Folder XML parameters (on page 60) for a list of the parameters that can be used/specified in sub-folder elements.
jobs	Defines a sequence of job elements. See Job and SMART Folder XML parameters (on page 60) for a list of the parameters that can be used or specified in job elements.

Job and SMART Folder XML parameters

The following table describes the job and SMART_folder XML parameters:

Parameter	Description
active_from	Indicates the start of a period of time during which the job or SMART Folder can be ordered.
active_till	Indicates the end of a period of time during which the job or SMART Folder can be ordered.
adjust_condition	Determines whether to ignore prerequisite conditions normally set by predecessor jobs if the relevant predecessor jobs are not scheduled.This parameter is relevant only for jobs in a SMART Folder. Optional.
and_or	Indicates the relationship between specified Days parameter values and Week days parameter values.
application	Provides a logical name for sorting groups of jobs. This parameter is used to supply a common descriptive name to a set of related job groups. The jobs do not necessarily have to run at the same time.
application_cm_version	Indicates the version of external application Control Module (such as SAP), which is installed in the Control-M installation. This is specified together with **application_form**, **application_type**, and **application_version** elements.
application_form	Specifies a predefined set of external application parameters that display in external Application panel of the Control-M Workload Automation Properties pane.
application_type	Indicates the type of external application (such as SAP) on which the external application job runs.
application_version	Indicates the version of the external application (such as SAP) on which the external application job runs.
arch_max_days	(Control-M for z/OS only) Defines the maximum number of days to retain the **SYSDATA** archive data set for jobs that ended **NOTOK**.
arch_max_runs	(Control-M for z/OS only) Defines the maximum number of job runs to retain the **SYSDATA** archive data set for jobs that ended **NOTOK**.

Parameter	Description
auto_archive	(Control-M for z/OS only)Determines whether SYSDATA (job output) must be archived.
captures (on page 126)	Indicates the parameters for job capture. See XML parameters for CAPTURES.
command	(Optional) Defines the command string supplied when the job Task Type (the **task_type** element) is **Command**.
confirm_flag	Specifies whether user confirmation is required before the job is submitted for execution.
control_resources (on page 122)	(From Forecast only) Indicates the resources required by the job during execution and the type of control (shared or exclusive) the job requires over each resource. The Control Resources parameter is used to control parallel execution of jobs.
count_cyclic_from	Indicates whether the interval between runs of a cyclic job or until the start of a rerun job is measured from the start or the end of the previous job run.
conf_cal	Indicates the name of a Control-M calendar that is used to validate scheduling dates. A shift value can be used to indicate how to handle jobs that are scheduled for a non-working day in the calendar.
created_by	Indicates the Control-M/EM user who defined the job.
critical	Determines whether the job is a critical-path job in Control-M, which ensures resources allocation order.
ctld_category	(Control-M for z/OS only) Defines the name of a Control-D Report Decollating Mission category to be scheduled whenever the job is run.
cyclic	Indicates that the job must run at a designated time, interval of time.

Parameter	Description
cyclic_type	Indicates how the intervals for running the job are specified, if job is cyclic (**cyclic** equal **yes**). Valid values are: ▪ **interval**: Job is run at fixed interval. See **rerun_interval**. ▪ **interval_sequence**: Job is run according to a list of time periods. See **interval_sequence**. ▪ **specific_times**: Job is run according to a list of specific times. See **specific_times**. Only for jobs running in Control-M/Server version 6.4.01 and above and Control-M for z/OS version 7.0.01 and above.
dates	Defines a sequence of **date,** which indicates a specific date (either mmdd or ddmm format, depending on the site standard), on which the job should be scheduled. Control-M for z/OS: A maximum of 12 date elements can be specified.
days_cal	Indicates the name of a user-defined calendar containing a list of days of the month, used with Month Days, to determine a set of working days.
days_due_out_offset	Defines the number of days that job execution can be extended after the ODAT. Only for jobs running in Control-M for z/OS version 6.2.00 and above.
dayskeepinnotok	Enables you to specify a minimum period to keep the SMART folder (and jobs) in the Active Jobs database after the folder is set to NOT OK.
description	Defines the text description of the job.
doc_lib	Defines the name of the directory/library containing the job documentation file.
doc_member	Defines the name of the file containing job documentation.
in_conditions (on page 122)	(From Forecast only) Specifies prerequisite conditions that must be satisfied before the job is submitted for execution. The In Conditions parameter makes the submission of the job dependent on the existence of one or more prerequisite conditions.

Parameter	Description
instream_jcl	Defines a script exactly as it would be specified in a terminal for the specific computer and is part of the job definition.
interval_sequence (on page 126)	Defines a sequence of **interval_item**. This parameter is relevant only for jobs running in Control-M/Server version 6.4.01 and above and Control-M for z/OS version 7.0.01 and above.
job_name	Defines the name of the job.
job_rule_based_cals	Defines a sequence of job_rule_based_cal, which refers to the name of the rule based calendar.
keep_active	Determines the number of extra days (beyond the original scheduling date) that the job is allowed to remain in the Active Jobs database while awaiting execution. If the job still has not run after the specified number of days, the job is removed from the Active Jobs database.
file_path	For non-z/OS jobs, File Path indicates the location of the file that contains the script. For z/OS jobs, Member Library indicates the location of the Member that contains the JCL, started task procedure, or Warning message.
file_name	Defines the name of the folder. In the Properties pane this parameter indicates the folder where the job belongs.
min_pds_tracks	Minimum number of free partitioned data set tracks required by the library specified for the Partition Data Set parameter.
month_days	Indicates the days of the month on which the job should be scheduled for processing.
multiagent	Specifies that job submission details be broadcast to all agents within a defined Host Group. All available agents in the Host Group run an identical job, and each such job has a unique Order ID.
host_id	Defines the Host ID of the host on which the job was recently run. Not for Control-M for z/OS jobs.

Parameter	Description
on_do_statements (on page 124)	Defines a sequence of on_do_statements, which consist of the following: ▪ **on_statements:** A sequence of on_statements XML parameters (on page 115) ▪ **do_statements:** A sequence of do_statements type XML parameters (on page 117)
out_conditions	(From Forecast only) Specifies prerequisite conditions to be added or deleted after the job completes with a completion status of OK.
override_path	Specifies a temporarily-modified job script file without changing the original script file in the File Path/Member library and without changing the scheduling order of a folder.
pds	(Control-M for z/OS only) Defines the name of a partitioned data set to check for free space. If the Partition Data Set has fewer than the minimum number of required free tracks (as specified for the Minimum number of tracks parameter), the job is executed.
prevent_nct2	(Control-M for z/OS only) Performs data set cleanup before the original job run.
priority	Determines the order of job processing by Control-M in the Active Jobs database. String.
quantitative_resources (on page 123)	(From Forecast only) Indicates the name and quantity of Quantitative resources required by the job.
removeatonce	Indicates that all jobs in the folder are not removed automatically from the Active Jobs database. Instead jobs wait for the folder to complete and are removed at the same time as the folder.
request_nje	(Control-M for z/OS only) Defines the node in the JES network where the job executes.
rerun_interval	Specifies the length of time to wait between reruns of a job or between cyclic runs of a job.
rerun_max	Determines the maximum number of reruns that can be performed for the job.
rerun_member	(Control-M for z/OS only) Defines the name of the JCL member to use when the job automatically reruns.

Parameter	Description
reten_days	(Control-M for z/OS, only) Determines the number of days to retain the job in the History Jobs file. For z/OS jobs only. String.
reten_gen	(Control-M for z/OS, only) Maximum number of generations of the job to keep in the History Jobs file. For z/OS jobs, only.Defines the maximum number of generations of the job to keep in the History Jobs file. String.
retro	Indicates if the job should be scheduled for possible execution after its original scheduling date has passed.
run_as	Identifies the user name with the authorization to execute the job. This parameter is used by the Control-M security mechanism.
SAC	(Control-M for z/OS only) Determines whether to adjust the logical date for a job converted from a scheduling product other than Control-M**.**
schedule_environment	Indicates the JES2 workload management scheduling environment that is to be associated with the job.
smart_folder_rule_based_cals	Defines a sequence of rule_based_cal XML parameters (on page 70)
shift	Enables you to schedule the job if the date is not confirmed. (Option) Valid values are: ▪ **ignore_job:** Do not shift the job to a different date. The job is not scheduled. ▪ **next_day:** Shift to the next working date. ▪ **prev_day:** Shift to the previous working date. ▪ **no_confcal:**. Tentatively schedule the job for the current day (even if not a working day). Additional shifting may or may not be performed, depending on the value indicated in the shift_num parameter.
shift_num	Defines the number of working days that a job can be shifted. Values from -62 to 62 can be entered. This function is also called Extended Shift.
shouts (on page 125)	Defines a sequence of **shout**.

Parameter	Description
specific_times (on page 126)	Defines a sequence of **specific_time**. Only for jobs running in Control-M/Server version 6.4.01 and above and Control-M for z/OS version 7.0.01 and above.
statistic_calendar	Name of the Control-M periodic calendar within which statistics relating to the job are collected.Defines the name of the Control-M periodic calendar within which statistics relating to the job are collected. Only for jobs running in Control-M for z/OS version 6.2.00 and above.
step_ranges (on page 124)	Defines a sequence of **step_range**.
sub_application	Defines the name of the sub application to which the job belongs. Used as a descriptive name for related groups of jobs.
sys_db	(Control-M for z/OS only)Indicates that a single data set is used for archiving the SYSDATA of all jobs until it is full, when another data set is started. Valid values: ▪ **yes:** Single data set created for the SYSDATA of each job run. ▪ **no:** Separate data set created for the SYSDATA of each job run.
output_from_class	(Control-M for z/OS only) Limits the output handling operation to only outputs from the specified class.

Parameter	Description
output_option	Enables output handling options. Optional. Valid values (non-Control-M for z/OS): ▪ **copy** ▪ **delete** ▪ **move** ▪ **release** Valid values (Control-M for z/OS): ▪ **copy** ▪ **delete** ▪ **move** ▪ **release** ▪ **change_class**
output_parameter	Defines **output_option** values, which require you to supply the following additional information (such as **Copy**, **NewDest**): ▪ If the output_option element is change_class, the output_parameter value corresponds to the new class name. ▪ If the output_option element is copy, the output_parameter value corresponds to the destination file name. ▪ If the output_option element is move, the output_parameter value corresponds to the new destination for the file.
rule_based_cal_relationship	Indicates the relationship (AND/OR) between Rule Based Calendar criteria and basic scheduling criteria in the job processing definition (that is, whether either set of criteria, or both sets of criteria, must be satisfied).
system_affinity	(Control-M for z/OS jobs only) Indicates the identity of the system in which the job must be initiated and executed (in JES2). Indicates the identity of the processor on which the job must execute (in JES3).
rule_based_cal_relationship	Indicates the relationship (AND/OR) between Rule Based Calendar criteria and basic scheduling criteria in the job processing definition (that is, whether either set of criteria, or both sets of criteria, must be satisfied).

Parameter	Description
task_type	Defines the type of the job (task) to be performed by Control-M. Valid values **Microsoft Windows and UNIX** ▪ job ▪ command ▪ dummy ▪ detached ▪ external **Control-M for z/OS** ▪ job ▪ task ▪ cyclic_job ▪ emergency_job ▪ emergency_cyclic_job ▪ cyclic_task ▪ emergency_task ▪ emergency_cyclic_task
time_due_out	Defines the time that the job is expected to finish.
time_from	Indicates the earliest time for submitting the job.
time_from_days_offset	Defines the number of days after the original scheduling date of the job during which execution of the job can begin. Only for jobs running in Control-M for z/OS version 6.2.00 and above.
time_until	Indicates the latest time for submitting the job.
time_until_days_offset	Defines the number of days after the original scheduling date of the job during which execution of the job can end. **NOTE:** Only for jobs running in Control-M for z/OS version 6.2.00 and later
time_zone	Indicates the global time zone used to calculate the interval for time-related conditions.

Parameter	Description
tolerance	Defines the maximum delay in minutes permitted for late submission when selecting a specific time. **NOTE:** Only for jobs running in Control-M/Server version 6.4.01 and later and Control-M for z/OS version 7.0.01 and later.
variable_assignments	Defines a sequence of variable_assignment XML parameters.
weeks_cal	Defines a name of a user-defined, week-based calendar used together with parameter Week Days to specify a set of working days.
week_days	Indicates the days of the week on which the job should be scheduled for processing.
JAN	Indicates whether to run the job in the relevant month. Valid values are: ▪ yes ▪ no
FEB	
MAR	
APR	
MAY	
JUN	
JUL	
AUG	
SEP	
OCT	
NOV	
DEC	

rule_based_cal XML parameters

The following figure describes the rule_based_cal XML parameters:

Parameter	Description
rule_base_cal_name	Defines the name of the Rule-Based Calendar.
keep_active	Defines the maximum number of days that the job can wait to be executed after its original scheduling date has passed.
and_or	Indicates the relationship between Month Days parameter values and Week Days parameter values. Optional.
days_cal	Defines the name of a user-defined calendar containing a list of days of the month, used with Month Days, to determine a set of working days.
weeks_cal	Defines the name of a user-defined, week-based calendar used together with parameter Week Days to specify a set of working days.
conf_cal	Defines the calendar used to confirm job scheduling dates.
retro	Indicates whether the job should be scheduled for possible execution after its original scheduled date has passed.
shift	Defines when to schedule the job if the date is not confirmed. (Option) Valid values are: ▪ **ignore_job**: Do not shift the job to a different date. The job is not scheduled. ▪ **next_day**: Enables you to shift to the next working date. ▪ **prev_day**: Enables you to shift to the previous working date. ▪ **no_confcal:** Enables you to tentatively schedule the job for the current day (even if not a working day). Additional shifting may or may not be performed, depending on the value indicated in the shift_num parameter.
shift_num	Defines the number of working days that a job can be shifted. Values from -62 to 62 can be entered. This function is also called Extended Shift.
active_from	Indicates the start of a period of time during which the job or SMART Folder can be ordered.
active_till	Indicates the end of a period of time during which the job or SMART Folder can be ordered.
month_days	Indicates the days of the month on which the job should be scheduled for processing.

Parameter	Description
week_days	Indicates the days of the week on which the job should be scheduled for processing.
dates	Defines a sequence of **date**, which indicates a specific date, in either mmdd or ddmm format (depending on the site standard), on which the job should be scheduled. **Control-M for z/OS**: A maximum of 12 **date** elements can be specified.
JAN	Indicates whether to run the job in this month. Valid values are: ▪ yes ▪ no
FEB	
MAR	
APR	
MAY	
JUN	
JUL	
AUG	
SEP	
OCT	
NOV	
DEC	

response_def_create_folder XML parameters

The following table describes the response_def_create_folder XML Parameters:

Parameter	Description
control_m	Indicates the name of the Control-M installation that processes the request. String.
folder_name	Indicates the name of the outermost folder. String.
folder_library	(Control-M for z/OS only) Indicates the name of the library in which the table is located. String.

For examples of responses, see Create folder definitions examples (on page 72). For error codes, see Create job/SMART Folder definitions request errors (Major code 412) (on page 208).

NOTE: XML parameters for fault_def_create_smart_folder and a sample fault response are described in Fault Response (on page 185).

Create folder definitions examples

The following topic describes examples of creating folders, jobs and SMART folders together with responses received from Control-M/EM:

Request to create SMART Folder and sub folder example

The following example is request is to create a outer-most SMART Folder with a job and a sub folder in the first level which also contains a job:

```
<?xml version="1.0" encoding="iso-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

  <SOAP-ENV:Body>

    <ctmem:request_def_create_folder
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">


<ctmem:user_token>115B95A7FA4B787B924A2BEA8CE3121A</ctmem:user_token>

      <ctmem:SMART_folder>
```

```
        <ctmem:control_m>my900</ctmem:control_m>
        <ctmem:folder_name>apiTestSMARTFolder</ctmem:folder_name>
        <ctmem:order_method>SYSTEM</ctmem:order_method>
        <ctmem:folder_attributes>
          <ctmem:application>apiWinApp</ctmem:application>
          <ctmem:sub_application>apiGroup</ctmem:sub_application>
          <ctmem:created_by>emuser</ctmem:created_by>
          <ctmem:smart_folder_rule_based_cals>
              <ctmem:rule_based_cal>
              <ctmem:rule_based_cal_name>rule_based_cal1</ctmem:rule_based
_cal_name>
               <ctmem:month_days>ALL</ctmem:month_days>
               <ctmem:MAR>yes</ctmem:MAR>
            </ctmem:rule_based_cal>
          </ctmem:smart_folder_rule_based_cals>
        </ctmem:folder_attributes>
        <ctmem:sub_folders>
          <ctmem:sub_folder>
            <ctmem:folder_name>subFolder_1</ctmem:folder_name>
            <ctmem:folder_attributes>
              <ctmem:application>apiWinApp</ctmem:application>
              <ctmem:group>apiGroup</ctmem:group>
              <ctmem:run_as>controlm</ctmem:run_as>
              <ctmem:created_by>emuser</ctmem:created_by>
              <ctmem:sub_folder_rule_based_cals>
                <ctmem:sub_rule_based_cal>
<ctmem:rule_based_cal_name>rule_based_cal1</ctmem:rule_based_cal_name>
                </ctmem:sub_rule_based_cal>
              </ctmem:sub_folder_rule_based_cals>
            </ctmem:folder_attributes>
<ctmem:jobs>
              <ctmem:job>
                <ctmem:job_name>In-SubFolder_1</ctmem:job_name>
                <ctmem:mem_name>In-SubFolder_1</ctmem:mem_name>
                <ctmem:task_type>command</ctmem:task_type>
```

```
                <ctmem:application>apiWinApp</ctmem:application>
                <ctmem:sub_application>apiGroup</ctmem:sub_application>
                <ctmem:command>ls -l</ctmem:command>
                <ctmem:run_as>controlm</ctmem:run_as>
                <ctmem:created_by>emuser</ctmem:created_by>
                <ctmem:job_rule_based_cals>
                  <ctmem:job_rule_based_cal>
<ctmem:rule_based_cal_name>rule_based_cal1</ctmem:rule_based_cal_name>
                  </ctmem:job_rule_based_cal>
                </ctmem:job_rule_based_cals>
              </ctmem:job>
            </ctmem:jobs>
          </ctmem:sub_folder>
        </ctmem:sub_folders>
<ctmem:jobs>
          <ctmem:job>
            <ctmem:job_name>In-Smart-Folder</ctmem:job_name>
            <ctmem:mem_name>INGROUP</ctmem:mem_name>
            <ctmem:mem_lib>INGROUP</ctmem:mem_lib>
            <ctmem:task_type>dummy</ctmem:task_type>
            <ctmem:application>apiWinApp</ctmem:application>
            <ctmem:sub_application>apiGroup</ctmem:sub_application>
            <ctmem:run_as>controlm</ctmem:run_as>
            <ctmem:created_by>emuser</ctmem:created_by>
          </ctmem:job>
        </ctmem:jobs>
      </ctmem:SMART_folder>
    </ctmem:request_def_create_folder>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

response_def_create_folder XML example

The following example describes a response:

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <ctmem:response_def_create_folder
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
      <ctmem:control_m>my900</ctmem:control_m>
      <ctmem:folder_name>apiTestSMARTFolder</ctmem:folder_name>
    </ctmem:response_def_create_folder>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Request to create a folder with a job example

The following describes a request to create a folder with a job:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <ctmem:request_def_create_folder
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

<ctmem:user_token>115B95A7FA4B787B924A2BEA8CE3121A</ctmem:user_token>
      <ctmem:folder>
        <ctmem:control_m>my900</ctmem:control_m>
        <ctmem:folder_name>apiTestFolder</ctmem:folder_name>
        <ctmem:order_method>SYSTEM</ctmem:order_method>
        <ctmem:jobs>
          <ctmem:job>
            <ctmem:job_name>In-Folder-1</ctmem:job_name>
            <ctmem:mem_name>In-Folder-1</ctmem:mem_name>
            <ctmem:task_type>dummy</ctmem:task_type>
            <ctmem:application>apiWinApp</ctmem:application>
            <ctmem:sub_application>apiGroup</ctmem:sub_application>
            <ctmem:run_as>controlm</ctmem:run_as>
            <ctmem:created_by>emuser</ctmem:created_by>
            <ctmem:month_days>ALL</ctmem:month_days>
            <ctmem:MAR>yes</ctmem:MAR>
```

```
          </ctmem:job>
        </ctmem:jobs>
      </ctmem:folder>
    </ctmem:request_def_create_folder>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response to failure to create a folder with job definitions example

The following example describes a response to the failure to create a folder with job definitions:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>SOAP-ENV:Server</faultcode>
      <faultstring>Error response from EM Server.</faultstring>
      <detail>
        <ctmem:fault_def_create_folder
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
          <ctmem:error_list ctmem:highest_severity='Error'>
            <ctmem:error ctmem:major='412' ctmem:minor='1'
ctmem:severity='Error'>
              <ctmem:error_message>Create jobs definitions failed, inavlid
              params.</ctmem:error_message>
            </ctmem:error>
            <ctmem:error ctmem:major='412' ctmem:minor='14'
ctmem:severity='Error'>
              <ctmem:error_message>Create jobs definitions validation error:
Job[1]: Field: Owner -
              Error: EM50011E: The field must have a
value.</ctmem:error_message>
            </ctmem:error>
          </ctmem:error_list>
        </ctmem:fault_def_create_folder>
      </detail>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
```

```
</SOAP-ENV:Envelope>
```

Request to create a folder fails (folder exists) example

The following example describes a request to create a folder that fails, although the folder exists:

```
<?xml version="1.0" encoding="iso-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

  <SOAP-ENV:Body>

    <ctmem:request_def_create_folder
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">


<ctmem:user_token>115B95A7FA4B787B924A2BEA8CE3121A</ctmem:user_token>

      <ctmem:folder>

        <ctmem:control_m>my900</ctmem:control_m>

        <ctmem:folder_name>apiTestFolder</ctmem:folder_name>

        <ctmem:order_method>SYSTEM</ctmem:order_method>

        <ctmem:jobs>

          <ctmem:job>

            <ctmem:job_name>In-Folder-2</ctmem:job_name>

            <ctmem:mem_name>In-Folder-2</ctmem:mem_name>

            <ctmem:task_type>dummy</ctmem:task_type>

            <ctmem:application>apiWinApp</ctmem:application>

            <ctmem:sub_application>apiGroup</ctmem:sub_application>

            <ctmem:run_as>controlm</ctmem:run_as>

            <ctmem:created_by>emuser</ctmem:created_by>

            <ctmem:month_days>ALL</ctmem:month_days>

            <ctmem:MAR>yes</ctmem:MAR>

          </ctmem:job>

        </ctmem:jobs>

      </ctmem:folder>

    </ctmem:request_def_create_folder>

  </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Response to failure to create a folder (folder exists) example

The following example describes a response to a failure to create a folder, although the folder exists:

```
<?xml version="1.0" encoding="iso-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

  <SOAP-ENV:Body>

    <SOAP-ENV:Fault>

      <faultcode>SOAP-ENV:Server</faultcode>

      <faultstring>Error response from EM Server.</faultstring>

      <detail>

        <ctmem:fault_def_create_folder
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

          <ctmem:error_list ctmem:highest_severity='Error'>

            <ctmem:error ctmem:major='412' ctmem:minor='7'
ctmem:severity='Error'>

              <ctmem:error_message>Folder already
exist.</ctmem:error_message>

            </ctmem:error>

          </ctmem:error_list>

        </ctmem:fault_def_create_folder>

      </detail>

    </SOAP-ENV:Fault>

  </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Add jobs to folder definitions

This request adds jobs to an existing SMART folder, sub-folder, or regular folder.

The following topics describe the request to add jobs to folder definitions parameters, the response from Control-M/EM, together with examples:

request_def_add_jobs XML parameters

The following table describes the request_def_add_jobs XML Parameters:

Parameter	Description
user_token	Defines the serial identification number supplied to the user during registration. String.
control_m	Defines the name of the Control-M installation that processes the request. String.
folder_name	Defines the name of the folder. String. Full slash (/) separates the parent folder name. For example, to add job X to an existing sub-folder C, that resides in sub-folder B, that resides in SMART outermost folder A, specify "A/B/C" in the folder_name.
folder_library	(Control-M for z/OS only) Defines the name of the library in which the table is located. String.
jobs	Defines a sequence of **job**. For a list of the parameters for job, refer to Job and SMART Folder XML parameters

For example of requests, see request_def_add_folder XML example (sub folder to sub folder) (on page 79) and request_def_add_jobs XML example (job to sub folder) (on page 81).

request_def_add_folder XML example (sub folder to sub folder)

The following example is a successful request is to add a sub folder with one job to an existing sub folder:

```
<?xml version="1.0" encoding="iso-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

  <SOAP-ENV:Body>

    <ctmem:request_def_add_folder
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">


<ctmem:user_token>D71B430E6A1B00071D4BE91A64E41AA4</ctmem:user_token>

      <ctmem:parent_folder>

        <ctmem:control_m>my900</ctmem:control_m>


<ctmem:folder_name>apiTestSMARTFolder/subFolder_1</ctmem:folder_name>

      </ctmem:parent_folder>

      <ctmem:sub_folder>
```

```
        <ctmem:folder_name>subFolder_2</ctmem:folder_name>
        <ctmem:folder_attributes>
          <ctmem:application>apiWinApp</ctmem:application>
          <ctmem:sub_application>apiGroup</ctmem:sub_application>
          <ctmem:run_as>controlm</ctmem:run_as>
          <ctmem:created_by>emuser</ctmem:created_by>
          <ctmem:sub_folder_rule_based_cals>
            <ctmem:sub_rule_based_cal>

<ctmem:rule_based_cal_name>rule_based_cal2</ctmem:rule_based_cal_name>
            </ctmem:sub_rule_based_cal>
          </ctmem:sub_folder_rule_based_cals>
        </ctmem:folder_attributes>
        <ctmem:jobs>
          <ctmem:job>
            <ctmem:job_name>In-SubFolder_2_job1</ctmem:job_name>
            <ctmem:mem_name>In-SubFolder_2_job1</ctmem:mem_name>
            <ctmem:task_type>command</ctmem:task_type>
            <ctmem:application>apiWinApp</ctmem:application>
            <ctmem:sub_application>apiGroup</ctmem:sub_application>
            <ctmem:command>ls -l</ctmem:command>
            <ctmem:run_as>controlm</ctmem:run_as>
            <ctmem:created_by>emuser</ctmem:created_by>
            <ctmem:job_rule_based_cals>
              <ctmem:job_rule_based_cal>

<ctmem:rule_based_cal_name>rule_based_cal2</ctmem:rule_based_cal_name>
              </ctmem:job_rule_based_cal>
            </ctmem:job_rule_based_cals>
          </ctmem:job>
        </ctmem:jobs>
      </ctmem:sub_folder>
    </ctmem:request_def_add_folder>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

request_def_add_jobs XML example (job to sub folder)

The following example is a successful request to add a job to an existing sub-folder:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <ctmem:request_def_add_jobs
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

<ctmem:user_token>83FE7394E4D10220EF99C3D55B79EFB4</ctmem:user_token>
      <ctmem:control_m>my900</ctmem:control_m>

<ctmem:folder_name>apiTestSMARTFolder/subFolder_1</ctmem:folder_name>
      <ctmem:jobs>
        <ctmem:job>
          <ctmem:job_name>In-SubFolder_job2</ctmem:job_name>
          <ctmem:mem_name>In-SubFolder_job2</ctmem:mem_name>
          <ctmem:task_type>dummy</ctmem:task_type>
          <ctmem:application>apiWinApp</ctmem:application>
          <ctmem:sub_application>apiGroup</ctmem:sub_application>
          <ctmem:run_as>controlm</ctmem:run_as>
          <ctmem:created_by>emuser</ctmem:created_by>
        </ctmem:job>
        <ctmem:job>
          <ctmem:job_name>In-SubFolder_job3</ctmem:job_name>
          <ctmem:mem_name>In-SubFolder_job3</ctmem:mem_name>
          <ctmem:task_type>dummy</ctmem:task_type>
          <ctmem:application>apiWinApp</ctmem:application>
          <ctmem:sub_application>apiGroup</ctmem:sub_application>
          <ctmem:run_as>controlm</ctmem:run_as>
          <ctmem:created_by>emuser</ctmem:created_by>
        </ctmem:job>
      </ctmem:jobs>
    </ctmem:request_def_add_jobs>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

response_def_add_jobs XML parameters

The following table describes the response_def_add_jobs XML Parameters:

Parameters	Description
control_m	Indicates the name of the Control-M installation that processes the request. String.
folder_name	Indicates the name of the outermost folder. String.
folder_library	(Control-M for z/OS only) Indicates the name of the library in which the folder is located. String.
number_of_jobs_added	Indicates the number of jobs added to the folder.

NOTE: XML parameters for fault_def_add_jobs, and a sample fault response are described in Fault Response (on page 185).

Add folder to folder definitions

This request adds a sub-folder to an existing SMART folder or sub-folder.

The following topics describe the request to add folder to folder definition parameters, the response from Control-M/EM, together with examples:

- request_def_add_folder XML parameters (on page 83)
- request_def_add_folder XML example (sub folder to sub folder) (on page 79)
- response_def_add_folder XML parameters (on page 85)
- response_def_add_folder XML parameters example (on page 85)

request_def_add_folder XML parameters

The following table describes the request_def_add_folder XML Parameters:

Parameter	Description
user_token	Defines the serial identification number supplied to the user during registration. String.
parent_folder	Defines a folder wrapper. The folder is identified by the following elements: ▪ **control_m:** Name of the Control-M installation that processes the request. String. ▪ **folder_name:** Name of the folder. String. Full slash (/) separates the parent folder name. For example, to add job X to an existing sub-folder C, that resides in sub-folder B, that resides in SMART outermost folder A, specify "A/B/C" in the folder_name. ▪ **folder_library: (Control-M for z/OS only)** Name of the library in which the folder is located. String.
jobs	Defines a sequence of **job**. For a list of the parameters for job, refer to Job and SMART Folder XML parameters.

For an example of a request, see request_def_add_folder XML example (sub folder to sub folder) (on page 79).

request_def_add_folder XML example (sub folder to sub folder)

The following example is a successful request is to add a sub folder with one job to an existing sub folder:

```
<?xml version="1.0" encoding="iso-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <ctmem:request_def_add_folder
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

<ctmem:user_token>D71B430E6A1B00071D4BE91A64E41AA4</ctmem:user_token>
      <ctmem:parent_folder>
        <ctmem:control_m>my900</ctmem:control_m>

<ctmem:folder_name>apiTestSMARTFolder/subFolder_1</ctmem:folder_name>
      </ctmem:parent_folder>
      <ctmem:sub_folder>
```

```
        <ctmem:folder_name>subFolder_2</ctmem:folder_name>
        <ctmem:folder_attributes>
          <ctmem:application>apiWinApp</ctmem:application>
          <ctmem:sub_application>apiGroup</ctmem:sub_application>
          <ctmem:run_as>controlm</ctmem:run_as>
          <ctmem:created_by>emuser</ctmem:created_by>
          <ctmem:sub_folder_rule_based_cals>
            <ctmem:sub_rule_based_cal>

<ctmem:rule_based_cal_name>rule_based_cal2</ctmem:rule_based_cal_name>
            </ctmem:sub_rule_based_cal>
          </ctmem:sub_folder_rule_based_cals>
        </ctmem:folder_attributes>
        <ctmem:jobs>
          <ctmem:job>
            <ctmem:job_name>In-SubFolder_2_job1</ctmem:job_name>
            <ctmem:mem_name>In-SubFolder_2_job1</ctmem:mem_name>
            <ctmem:task_type>command</ctmem:task_type>
            <ctmem:application>apiWinApp</ctmem:application>
            <ctmem:sub_application>apiGroup</ctmem:sub_application>
            <ctmem:command>ls -l</ctmem:command>
            <ctmem:run_as>controlm</ctmem:run_as>
            <ctmem:created_by>emuser</ctmem:created_by>
            <ctmem:job_rule_based_cals>
              <ctmem:job_rule_based_cal>

<ctmem:rule_based_cal_name>rule_based_cal2</ctmem:rule_based_cal_name>
              </ctmem:job_rule_based_cal>
            </ctmem:job_rule_based_cals>
          </ctmem:job>
        </ctmem:jobs>
      </ctmem:sub_folder>
    </ctmem:request_def_add_folder>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

response_def_add_folder XML parameters

The following table describes the response_def_add_folder XML Parameters:

Parameters	Description
control_m	Indicates the name of the Control-M installation that processes the request. String.
folder_name	Indicates the full name of the outermost folder. String.
folder_library	(Control-M for z/OS only) Indicates the name of the library in which the folder is located. String.

For an example of a response, see response_def_add_folder XML parameters example (on page 85).

NOTE: XML parameters for fault_def_add_folder, and a sample fault response are described in Fault Response (on page 185).

response_def_add_folder XML parameters example

The following example describes a successful response:

```
<?xml version="1.0" encoding="iso-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

  <SOAP-ENV:Body>

    <ctmem:response_def_add_folder
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

      <ctmem:control_m>my900</ctmem:control_m>


<ctmem:folder_name>apiTestSMARTFolder/subFolder_1/subFolder_2</ctmem:folde
r_name>

    </ctmem:response_def_add_folder>

  </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Delete job definitions

This request deletes one or more jobs from a folder according to the user defined deletion criteria. SMART Folder, Folder or sub-folder entities are not deleted.

If no jobs are deleted according to the specified deletion criteria, a fault response with the appropriate message is returned. The following topics describe the request to delete job definitions, the response from Control-M/EM, together with examples:

request_def_delete_jobs XML parameters

The following table describes the request_def_delete_jobs XML parameters:

Parameter	Description
user_token	Defines the serial identification number supplied to the user during registration. String.
folder	Defines the Folder wrapper, which is identified by the following parameters: ▪ **control_m:** Defines the name of the Control-M installation that processes the request. String. ▪ **folder_name:** Defines the Name of the folder. String. Full slash (/) separates the parent folder name. For example, to add job X to an existing sub-folder C, that resides in sub-folder B, that resides in SMART outermost folder A, specify "A/B/C" in the folder_name. ▪ **folder_library:** (Control-M for z/OS only) Defines the name of the library in which the table is located. String.
delete_jobs_criterion	Defines the delete jobs criteria wrapper. String. Consists of filter parameters that enable specifying items to do the following: ▪ **include:** (mandatory) Filter that consists of a sequence of search_criterion elements. ▪ **exclude:** (optional) Filter that consists of a sequence of search_criterion elements. See include or exclude XML parameters (on page 87) for more information.

For an example of a request, see request_def_add_folder XML example (on page 88).

include or exclude XML parameters

The following table describes the include or exclude parameters:

Parameter	Description
search_criterion	The search criteria wrapper consists of a sequence of param elements, which defines the paramters used to build the search criteria. Elements of param are listed in the param XML parameters (on page 87). At least one **search_criterion** element must appear under the exclude element. The amount of **search_criterion** elements is unbounded. The relationship between search_criterion elements in one filter is OR.

param XML parameters

The following table describes param XML parameters:

Parameter	Description
name	Defines the name of the table definition parameter used as a search criteria. String. Mandatory. The valid values for the name parameter are as follows: ▪ **APPLICATION:** Name of the application to which the job's group belongs. ▪ **GROUP_NAME:** Name of the group to which the job belongs ▪ **FILE_NAME:** Name of the file that contains the job script ▪ **JOB_NAME:** Name of the job ▪ **DESCRIPTION:** Description of the job ▪ **CREATED BY**: Control-M/EM user who defined the job. This argument is used by the Control-M security mechanism and under certain circumstances, cannot be modified. See AuthorSecurity parameters in GUI Server parameters. ▪ **HOST_ID:** Host ID of the host on which the job was most recently run (not for MVS jobs). ▪ **FILE_PATH:** Name of the path the contains the job script file. For more information see General parameters.

Parameter	Description
operator	Defines the operator used in search criteria. String. Valid values: ▪ **EQ** ▪ **NE** ▪ **LT** ▪ **GT** ▪ **LIKE**
value	Defines the value used in search criteria. Any valid value of a job parameter. Wildcards and search patterns can be used in combination with LIKE operator. String. Mandatory.

NOTE: At least one param element should appear under a **search_criterion** element. The amount of param elements is unbounded. The relationship between param elements in the same search_criterion is AND. String

request_def_add_folder XML example

The following example describes a successful request:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

  <SOAP-ENV:Body>

    <ctmem:request_def_delete_jobs
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">


<ctmem:user_token>D71B430E6A1B00071D4BE91A64E41AA4</ctmem:user_token>

      <ctmem:folder>

        <ctmem:control_m>my900</ctmem:control_m>


<ctmem:folder_name>apiTestSMARTFolder/subFolder_1/subFolder_2</ctmem:folde
r_name>

      </ctmem:folder>

      <ctmem:delete_jobs_criterion>

        <ctmem:include>

          <ctmem:search_criterion>

            <ctmem:param>

              <ctmem:name>JOB_NAME</ctmem:name>
```

```
            <ctmem:operator>LIKE</ctmem:operator>
            <ctmem:value>*</ctmem:value>
          </ctmem:param>
        </ctmem:search_criterion>
      </ctmem:include>
      <ctmem:exclude>
        <ctmem:search_criterion>
          <ctmem:param>
            <ctmem:name>MEMNAME</ctmem:name>
            <ctmem:operator>EQ</ctmem:operator>
            <ctmem:value>apiMemName2</ctmem:value>
          </ctmem:param>
        </ctmem:search_criterion>
      </ctmem:exclude>
    </ctmem:delete_jobs_criterion>
  </ctmem:request_def_delete_jobs>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

response_def_delete_jobs XML parameters

The following table describes the response_def_delete_jobs XML parameters:

Parameter	Description
folder	Indicates a folder wrapper. See the folder parameters in request_def_delete_jobs XML parameters (on page 86).
deleted_jobs_number	Indicates the number of jobs that are deleted. String.

For an example of a response, see response_def_delete_jobs XML example (on page 89). For error codes: see Delete job definitions request errors (Major code 413) (on page 209).

NOTE: XML parameters for fault_def_delete_jobs, and a sample fault response are described in Fault Response (on page 185).

response_def_delete_jobs XML example

The following example describes a successful response:

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <ctmem:response_def_delete_jobs
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
      <ctmem:folder>
        <ctmem:control_m>my900</ctmem:control_m>

<ctmem:folder_name>apiTestSMARTFolder/subFolder_1/subFolder_2</ctmem:folde
r_name>
      </ctmem:folder>
      <ctmem:deleted_jobs_number>1</ctmem:deleted_jobs_number>
    </ctmem:response_def_delete_jobs>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Upload folder

Uploads an outermost folder. Only one folder can be uploaded per request. It is possible to force upload a folder using the optional force parameter.

The following topics describe the request to upload folder parameters, polling request parameters, the response and the polling response from Control-M/EM, together with examples:

request_def_upload_folder XML parameters

The following table describes the request_def_upload_folder XML parameters:

Parameter	Description
user_token	Defines the serial identification number supplied to the user during registration. String.
force_it	Forces the uploading of a folder. String. Optional.
folder	Defines the folder wrapper. The folder is identified by the following parameter elements: ▪ **control-m:** Name of the Control-M installation that processes the request. String. ▪ **folder_name:** Name of the outermost folder. String. ▪ **folder_library: Control-M for z/OS only**: Name of the library in which the table is located. String

For an example of a request, see request_def_upload_folder XML example (on page 91).

request_def_upload_folder XML example

The following example describes a successful request:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <ctmem:request_def_upload_folder
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

<ctmem:user_token>17C3BA2475FB34069257862D90FB2128</ctmem:user_token>
      <ctmem:force_it>no</ctmem:force_it>
      <ctmem:folder>
        <ctmem:control_m>my900</ctmem:control_m>
        <ctmem:folder_name>apiTestSMARTFolder</ctmem:folder_name>
      </ctmem:folder>
    </ctmem:request_def_upload_folder>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

response_def_upload_folder XML Parameters

The following table describes the response_def_upload_folder XML Parameter:

Parameter	Description
response_token	Used in a polling request.

For an example of a response, see response_def_upload_folder example (on page 92). For error codes: see Upload folder request errors (Major code 411) (on page 208).

NOTE: XML parameters for fault_def_upload_folder and fault_poll_def_upload_folder, as well as a sample fault response are described in Fault Response (on page 185).

response_def_upload_folder example

The following example describes a successful response:

```
<?xml version="1.0" encoding="iso-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

  <SOAP-ENV:Body>

    <ctmem:response_def_upload_folder
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

      <ctmem:status>OK</ctmem:status>

      <ctmem:response_token>6</ctmem:response_token>

    </ctmem:response_def_upload_folder>

  </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

request_poll XML parameters

The following table describes the request_poll_def_upload_table XML parameters:

Parameter	Description
user_token	Defines the serial identification number supplied to the user during registration. String.
response_token	Enables you to receive a token in the immediate response of response_def_upload_table.

For an example of a poll request, see request_poll_def_upload_folder XML parameters example (on page 93).

request_poll_def_upload_folder XML parameters example

The following example describes a successful request:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

  <SOAP-ENV:Body>

    <ctmem:request_poll_def_upload_folder
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">


<ctmem:user_token>17C3BA2475FB34069257862D90FB2128</ctmem:user_token>

      <ctmem:response_token>6</ctmem:response_token>

    </ctmem:request_poll_def_upload_folder>

  </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

response_poll_def_upload_folder XML parameters

The following table describes response_poll_def_upload_folder XML parameter:

Parameter	Description
status	Indicates status of polling. String. Valid values: ▪ **OK** ▪ **EXEC**

For an example of a polling response, see response_poll_def_upload_folder XML example (on page 93).

response_poll_def_upload_folder XML example

The following example describes a successful response:

```
<?xml version="1.0" encoding="iso-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

    <SOAP-ENV:Body>

        <ctmem:response_poll_def_upload_folder
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

            <ctmem:status>OK</ctmem:status>

        </ctmem:response_poll_def_upload_folder>
```

```
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Order

The following operations can be performed with jobs and SMART Folders using Order:

- Order a job: Enables you to enter a job into Active Jobs only when its scheduling criteria are met
- Force a job: Enables you to enter a job into Active Jobs even if scheduling criteria is not met
- Order a SMART Folder: Enables you to order all jobs in the folder when its scheduling criteria is met.
- Force a SMART Folder: Enables you to enter a SMART Folder into Active Jobs even if its scheduling criteria is not met.

The following topics describe the request to order_force XML parameters, polling request parameters, the response and polling response from Control-M/EM, together with examples:

- request_order_force XML parameters (on page 95)
- request_poll_order_force XML parameter (on page 98)
- response_order_force XML parameters (on page 98)
- response_poll_order_force XML parameters (on page 99)
- Order XML parameters examples (on page 100)

The mandatory parameters for ordering a job differ from the mandatory parameters that are specified when ordering or forcing a SMART Folder. Optional parameters can be supplied for both jobs and SMART Folders.

request_order_force XML parameters

The following table describes request_order_force XML parameters:

Parameter	Description
user_token	Defines the serial identification number supplied to the user during registration. String.
control_m	Defines the data center name. Mandatory for both jobs and Folders.String.
folder_library	(Control-M for z/OS only) Defines the name of the library in which the folder is located. String
folder_name	Defines the name of the folder. Mandatory for both jobs and folders. String.
folder_info	Defines a sequence of **folder_info**. See the folder_info_type XML parameters (on page 97)
force_it	Indicates whether the job is ordered or forced. Mandatory for both jobs and Folders. ▪ **no:** Orders job/folder ▪ **yes:** Forces job/folder
job_id	Defines the occurrence number of the job within the folder. String. Optional Used to identify a specific job, in case there are multiple occurrences of the same job name in the folder. If the value of the job_id parameter is blank, 0, or 1, the first occurrence of the job is used.
job_name	Defines the name of the job (mandatory). Must be left empty for folders. String.
max_returned_nodes	Enables you to limit the number of returned entities. Optional. **NOTE:** Should not exceed the value of the **EMAPIActiveJobsLoadLimit** system parameter.
order_date	Enables you to order the job with a specific date. Mandatory for both jobs and folders. Valid values: ▪ Numerical date (yyyymmdd format) ▪ **ODAT**

Parameter	Description
unique_flow	Determines if a flow in a folder is ordered uniquely. This is only if you are ordering a single folder created in version 8.0.00 and later. A unique suffix is added to every condition name. See Condition management. Valid values: ▪ Yes ▪ No
variable_assignments (on page 97)	Defines a sequence of **variable** XML parameters: ▪ **Name**: Name of the variable (mandatory if the variable_assignment element is specified. String. Name sequence. ▪ **Value**: Value of the variable expression. String.
wait_for_order_date	Indicates whether the job submission should wait for a specified order date (order_date), or be submitted as soon as the execution criteria for the job is satisfied. Valid values: ▪ **yes**: Wait for order_date ▪ **no** Do not wait for order_date (default) **NOTE:** Relevant for Control-M versions 6.2.01 or later.
with_hold	Enables you to hold all jobs immediately after they are ordered.

For an example of a request, see Order XML parameters examples (on page 100).

folder_info_type XML parameters

The following table describes info_type XML parameters:

Parameter	Description
into_folder	Indicates into which SMART Folder the job is placed. Optional for jobs. Not used for folders. ▪ **recent** ▪ **new** ▪ **standalone** ▪ **selected** ▪ **order ID of a SMART Folder**
folder_id	Serial number identifying the SMART Folder. Optional for jobs. Must be empty for folders. For more information on the folder_id parameter, see Job tracking (on page 156). String.
allow_dup	Allows duplicate jobs in a SMART Folder. Optional for jobs. Must be left empty for folders (accepts default). Valid values: ▪ **no:** Not allowed ▪ **yes:** Allowed (default)

variable_assignment XML parameters

The following table describes variable_assignment XML parameters:

Parameter	Description
name	Name of the variable. Mandatory, if the **variable_assignment** element is specified. String. Name sequence.
value	Value of the Variable expression. String.

response_order_force XML parameters

The following table describes response_order_force XML parameters:

Parameter	Description
status	Indicates a description of message content. String.
response_token	Used in the polling request.

For examples of responses, see Order XML parameters examples (on page 100). For Error codes, see Order or Force request errors (Major code 405) (on page 205).

NOTE: XML parameters for fault_order_force and fault_poll_order_force, as well as a sample fault response are described in Fault Response (on page 185).

request_poll_order_force XML parameter

The following table describes request_poll_order_force XML parameters:

Parameter	Description
user_token	Defines the serial identification number supplied to the user during registration. String.
response_token	Enables you to receive a token in the immediate response of a response_order_force.

For an example of a polling request, see Order XML parameters examples (on page 100).

response_poll_order_force XML parameters

The following table describes response_poll_order_force XML parameters:

Parameter	Description	
status	Describes the condition of the element that contains it (such as **Error**). String.	
jobs	A sequence of **job**, which includes the following parameters: ▪ **Status** ▪ **error_list** ▪ **error_list attribute** ▪ **Job_data:** An element that contains other parameters that describe the job. A sequence of **job_data,** which contains the following parameters: ▪ **rba:** Relative block address. String ▪ **order_id:** Serial number assigned to the job by Control-M Workload Automation installation. String. ▪ **file_name:** Name of the file that contains the job script. String. ▪ **job_name:** Name of the job. String. ▪ **is_folder:** Indicates whether the job is a member of a SMART Folder. Valid values: **no** (not a member of a SMART Folder) or **yes** (member of a SMART Folder) ▪ **ret_text:** Text describing the job run. String.	
error_list	A sequence of **error**. See Fault Response (on page 185).	
error_list attribute	highest_severity	Indicates the severity level of the most critical error included in the error list. If only one error is included, the severity for that error is displayed. String.

For an example of a polling response, see Order XML parameters examples (on page 100).

Order XML parameters examples

The following topic describes the examples for the order XML parameter as well as examples as to ordering or forcing a job in Windows or UNIX.

request_order_force XML example

The following example describes a successful request:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

  <SOAP-ENV:Body>

    <ctmem:request_order_force
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">


<ctmem:user_token>17C3BA2475FB34069257862D90FB2128</ctmem:user_token>

      <ctmem:force_it>yes</ctmem:force_it>

      <ctmem:control_m>my900</ctmem:control_m>

      <ctmem:folder_name>apiTestSMARTFolder</ctmem:folder_name>

      <ctmem:odate>ODAT</ctmem:odate>

    </ctmem:request_order_force>

  </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

response_order_force XML example

The following example describes a successful response:

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <ctmem:response_order_force
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
      <ctmem:status>OK</ctmem:status>
      <ctmem:response_token>10u</ctmem:response_token>
    </ctmem:response_order_force>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

request_poll_order_force XML example

The following example describes a successful request:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <ctmem:request_poll_order_force
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

<ctmem:user_token>17C3BA2475FB34069257862D90FB2128</ctmem:user_token>
      <ctmem:response_token>10u</ctmem:response_token>
    </ctmem:request_poll_order_force>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

response_poll_order_force XML example

The following example describes a successful response:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <ctmem:response_poll_order_force
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
      <ctmem:status>OK</ctmem:status>
      <ctmem:jobs>
        <ctmem:job>
```

```
          <ctmem:status>OK</ctmem:status>
          <ctmem:job_data>
            <ctmem:order_id>00000j</ctmem:order_id>
            <ctmem:job_name>apiTestSMARTFolder</ctmem:job_name>
            <ctmem:ret_text>Job ordered</ctmem:ret_text>
          </ctmem:job_data>
        </ctmem:job>
        <ctmem:job>
          <ctmem:status>OK</ctmem:status>
          <ctmem:job_data>
            <ctmem:order_id>00000k</ctmem:order_id>
            <ctmem:job_name>subFolder_1</ctmem:job_name>
            <ctmem:ret_text>Job ordered</ctmem:ret_text>
          </ctmem:job_data>
        </ctmem:job>
        <ctmem:job>
          <ctmem:status>OK</ctmem:status>
          <ctmem:job_data>
            <ctmem:order_id>00000l</ctmem:order_id>
            <ctmem:mem_name>In-SubFolder_1</ctmem:mem_name>
            <ctmem:job_name>In-SubFolder_1</ctmem:job_name>
            <ctmem:ret_text>Job ordered</ctmem:ret_text>
          </ctmem:job_data>
        </ctmem:job>
      </ctmem:jobs>
    </ctmem:response_poll_order_force>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Order a UNIX job example

The following describes a successful request to order a UNIX job:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
```

```
        <ctmem:request_order_force
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
            <ctmem:user_token>$USER_TOKEN$</ctmem:user_token>
            <ctmem:force_it>no</ctmem:force_it>

<ctmem:control_m>UnixDc</ctmem:control_m><ctmem:job_id>2</ctmem:job_id>
            <ctmem:job_name>OrdSimJobU</ctmem:job_name>
            <ctmem:folder_name>OrdSimJobU</ctmem:folder_name>
            <ctmem:odate>20020522</ctmem:odate>
        </ctmem:request_order_force>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Force a UNIX job example

The following example describes a successful request to force a UNIX job:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <ctmem:request_order_force
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
            <ctmem:user_token>$USER_TOKEN$</ctmem:user_token>
            <ctmem:force_it>yes</ctmem:force_it>
            <ctmem:control_m>UnixDc</ctmem:control_m>
            <ctmem:job_id>2</ctmem:job_id>
            <ctmem:job_name>ForSimJobU</ctmem:job_name>
            <ctmem:folder_name>ForSimJobU</ctmem:folder_name>
            <ctmem:odate>20010522</ctmem:odate>
        </ctmem:request_order_force>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Force a UNIX job into a recent SMART Folder example

The following example describes a successful request to force a UNIX job into a recent SMART folder:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <ctmem:request_order_force
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
            <ctmem:user_token>$USER_TOKEN$</ctmem:user_token>
            <ctmem:force_it>yes</ctmem:force_it>
            <ctmem:control_m>UnixDc</ctmem:control_m>
            <ctmem:job_id>2</ctmem:job_id>
            <ctmem:job_name>FoInRecent</ctmem:job_name>
            <ctmem:folder_name>FoInSGJobU</ctmem:folder_name>
            <ctmem:odate>20010522</ctmem:odate>
            <ctmem:variable_assignments>
                <ctmem:variable_assignment>
                    <ctmem:name>Recent</ctmem:name>
                    <ctmem:value>1</ctmem:value>
                </ctmem:variable_assignment>
                <ctmem:variable_assignment>
                    <ctmem:name>A</ctmem:name>
                    <ctmem:value>1</ctmem:value>
                </ctmem:variable_assignment>
                <ctmem:variable_assignment>
                    <ctmem:name>A</ctmem:name>
                    <ctmem:value>2</ctmem:value>
                </ctmem:variable_assignment>
            </ctmem:variable_assignments>
            <ctmem:folder_info>
                <ctmem:into_folder>recent</ctmem:into_folder>
                <ctmem:allow_dup>no</ctmem:allow_dup>
            </ctmem:folder_info>
        </ctmem:request_order_force>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Force a UNIX job into a recent SMART Folder allowing duplication example

The following table describes a successful request to force a UNIX job into a ‘recent’ SMART Folder allowing duplication

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

    <SOAP-ENV:Body>

        <ctmem:request_order_force
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

            <ctmem:user_token>$USER_TOKEN$</ctmem:user_token>

            <ctmem:force_it>yes</ctmem:force_it>

            <ctmem:control_m>UnixDc</ctmem:control_m>

            <ctmem:job_id>6</ctmem:job_id>

            <ctmem:job_name>FoInRecDup</ctmem:job_name>

            <ctmem:folder_name>FoInSGJobU</ctmem:folder_name>

            <ctmem:odate>20010522</ctmem:odate>

            <ctmem:variable_assignments>

                <ctmem:variable_assignment>

                    <ctmem:name>RecentDup</ctmem:name>

                    <ctmem:value>2</ctmem:value>

                </ctmem:variable_assignment>

                <ctmem:variable_assignment>

                    <ctmem:name>A</ctmem:name>

                    <ctmem:value>2</ctmem:value>

                </ctmem:variable_assignment>

                <ctmem:variable_assignment>

                    <ctmem:name>B</ctmem:name>

                    <ctmem:value>3</ctmem:value>

                </ctmem:variable_assignment>

            </ctmem:variable_assignments>

            <ctmem:folder_info>

                <ctmem:into_folder>recent</ctmem:into_folder>

                <ctmem:allow_dup>yes</ctmem:allow_dup>

            </ctmem:folder_info>

        </ctmem:request_order_force>
```

```
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Force a sub-folder into a recent sub-folder example

The following example describes a successful request to force a sub-folder into a recent sub-folder:

```
 <?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
      <SOAP-ENV:Body>
            <ctmem:request_order_force
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
                  <ctmem:user_token>$USER_TOKEN$</ctmem:user_token>
                  <ctmem:force_it>yes</ctmem:force_it>
                  <ctmem:control_m>my900</ctmem:control_m>
                  <ctmem:job_name>sub2</ctmem:job_name>
                  <ctmem:folder_name>smart1/sub1</ctmem:folder_name>
                  <ctmem:odate>ODAT</ctmem:odate>
                  <ctmem:folder_info>
                        <ctmem:into_folder>recent</ctmem:into_folder>
                        <ctmem:allow_dup>yes</ctmem:allow_dup>
                  </ctmem:folder_info>
            </ctmem:request_order_force>
      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Job creation

Creates a job or SMART Folder and inserts the job into Active Jobs. You can create the following:

- Regular job
- SMART Folder
- Job in a SMART folder

You can indicate which SMART Folder to associate the job by specifying the folder name and folder ID (**folder_id**) for the parent SMART Folder in the folder parameter of the job specification.

The following topics describe the job creation request parameters, polling request parameters, the response and polling response from Control-M/EM, together with examples:

- request_create_aj XML parameters (on page 107)
- response _create_aj XML parameters (on page 127)
- request_poll_create_aj XML parameters (on page 128)
- response_poll_create_aj XML parameters (on page 128)
- Job creation examples (on page 129)

If the folder is not already ordered, it can be created using separate request parameters and before creating the dependent job. The folder name is specified when creating the new folder, and the necessary information (folder name, order_id, and RBA) can be obtained from the response to this request for use when populating the folder with jobs.

request_create_aj XML parameters

The following table describes request_create_aj XML parameters:

Parameter	Description
user_token	Defines a serial identification number supplied to the user during registration. String.
control-m	Defines the data center name. String. Mandatory.
active_job	Defines the specification for a single job. The parameters of the job are included as elements between the opening and closing **active_job** tags. Contains no attributes. See active_job XML parameters (on page 108) for more information.

For an example of a request, see Job creation examples (on page 129).

active_job XML parameters

The following table describes active_job XML Parameters:

Parameter	Description
adjust_condition	Determines whether to ignore prerequisite conditions normally set by predecessor jobs if the relevant predecessor jobs are not scheduled. (SMART folders only) Optional.
application	Provides a logical name for sorting groups of jobs. This parameter is used to supply a common descriptive name to a set of related job groups. The jobs do not necessarily have to run at the same time.
application_cm_version	Indicates the version of external application Control Module (such as SAP), which is installed in the Control-M installation. This is specified together with **application_form**, **application_type**, and **application_version** elements.
application_form	Specifies a predefined set of external application parameters that display in the External Application panel of the Control-M Workload Automation Properties pane.
application_type	Indicates the type of external application (such as SAP) on which the external application job runs.
application_version	Indicates the version of the external application (such as SAP) on which the external application job runs.
arch_max_days	(Control-M for z/OS only) Defines the maximum number of days to retain the **SYSDATA** archive data set for jobs that ended **NOTOK**.
arch_max_runs	(Control-M for z/OS only) Defines the maximum number of job runs to retain the **SYSDATA** archive data set for jobs that ended **NOTOK**.
auto_archive	(Control-M for z/OS only) Determines whether SYSDATA (job output) must be archived.
captures (on page 126)	Indicates the parameters for job capture. See XML parameters for CAPTURE.
command	(Optional) Indicates an operating system command line entry to be submitted as a job. Use this parameter to specify an operating system command to execute by Control-M. The command must be specified exactly as it would be specified in a terminal for the specific computer.
confirm_flag	Specifies whether user confirmation is required before the job is submitted for execution.

Parameter	Description
control_resources	(From Forecast only) Indicates the resources required by the job during execution and the type of control (shared or exclusive) the job requires over each resource. The Control Resources parameter is used to control parallel execution of jobs. A sequence of control_resources XML parameters (on page 122).
count_cyclic_from	Indicates whether the interval between successive runs of a cyclic job is calculated from the start or the end of the previous job run.
critical	Determines whether the job is a critical-path job in Control-M, which ensures resources allocation order.
ctb_step	Adds Control-M/Analyzer steps as the first and/or last step of the job's execution. A sequence of ctb_step XML parameters (on page 121).
Cyclic	Indicates that the job must run at a designated time, interval of time.
cyclic_type	Determines the type of cyclic job: ▪ **interval**: Job is run at fixed interval. See **rerun_interval.** ▪ **interval_sequence**: Job is run according to a list of time periods. See **interval_sequence**. ▪ **specific_times**: Job is run according to a list of specific times. See **specific_times**. **NOTE:** This parameter is relevant only for jobs running in Control-M/Server version 6.4.01 and later and Control-M for z/OS version 7.0.01 and later.
days_due_out_offset	Defines the number of days that job execution can be extended after the ODAT
description	Text description of the job.
doc_lib	For a z/OS job, Doc Library defines the name of the library where the Documentation (description) is saved. For a non-z/OS job, Doc Path defines the name of the file path where the Documentation is saved..
doc_member	For a z/OS job, defines the name of the member where the job Documentation (description) is saved. For a non-z/OS job, the Doc File is the name of the file where the job Documentation is saved.
folder_id	Defines the Order ID of the parent SMART Folder preceded by a leading zero. Control-M for Z/Os: The folder consists of RBA of the existing SMART folder. **NOTE:** After the SMART Folder has been created, note the value of the **order_id** parameter, which must be supplied to the **folder_id** job parameters.

Parameter	Description
in_condition	(From Forecast only) Specifies prerequisite conditions that must be satisfied before the job is submitted for execution. The In Conditions parameter makes the submission of the job dependent on the existence of one or more prerequisite conditions. See in_condition XML parameters (on page 122)
interval_sequence	A sequence of **interval_item**. See interval_sequence XML parameters (on page 126). **NOTE:** This parameter is relevant only for jobs running in Control-M/Server version 6.4.01 and later and Control-M for z/OS version 7.0.01 and later.
job_name	Defines the name of the job
keep_active	Determines the number of extra days (beyond the original scheduling date) that the job is allowed to remain in the Active Jobs database while awaiting execution. If the job still has not run after the specified number of days, the job is removed from the Active Jobs database. Integer.
file_path	For non-z/OS jobs, File Path indicates the location of the file that contains the script. For z/OS jobs, Member Library indicates the location of the Member that contains the JCL, started task procedure, or Warning message. String.
file_name	Indicates the name of the file that contains the job script, or for z/OS jobs, the name of a member that contains one of the following in relation to the job to be executed: ▪ The JCL of the job ▪ The started task procedure ▪ Warning messages
multiagent	Specifies that job submission details be broadcast to all agents within a defined Host Group. All available agents in the Host Group run an identical job, and each such job has a unique Order ID.
host_group	Defines the name of a Control-M/Agent computer, remote host computer, or host group where the job is submitted. Not for z/OS.
order_date	Original scheduling date of a job.
on_do_statements	A sequence of on_do_statements, which consist of the following: ▪ **on_statements:** A sequence of on_statements XML parameters (on page 115) ▪ **do_statements:** A sequence of do_statements type XML parameters (on page 117)

Parameter	Description
order_folder	Default or dummy folder to which you indicate the job belongs A folder is not necessary because jobs that are created with Control-M/EM API are inserted directly into Active Jobs. However, you may want to include a value for this parameter so that the job can be tracked during statistical analysis that uses **Folder** as a criterion.
order_library	Default or dummy folder library in which folder documentation is stored. A folder (and a folder library) are not necessary because jobs that are created with Control-M/EM API are entered directly into Active Jobs. However, you may want to include a value for this parameter so that the job can be tracked during statistical analysis that uses **Folder** or **Folder Library** as criteria. This parameter is specified only for z/OS jobs for which the **order_folder** element was also specified.
out_condition	(From Forecast only) Specifies prerequisite conditions to be added or deleted after the job completes with a completion status of OK. A sequence of out_condition XML parameters (on page 124)
run_as	Identifies the user name with the authorization to execute the job. This parameter is used by the Control-M security mechanism.
override_path	Specifies a temporarily-modified job script file without changing the original script file in the File Path/Member library and without changing the scheduling order of a folder.
pipes	Runs the job regardless of whether other jobs using the same Pipe as that specified in the definition of this job, are ready to run. See pipe XML parameters (on page 123)
prevent_nct2	(Z/OS only)Performs data set cleanup before the original job run.
priority	Determines the order of job processing by Control-M in the Active Jobs database. String.
quantitative_resources	(From Forecast only) Indicates the name and quantity of Quantitative resources required by the job. See quantitative_resource XML parameters (on page 123).
request_nje	(z/OS only) Defines the node in the JES network where the job executes. String.
rerun_interval	Specifies the length of time to wait between reruns of a job or between cyclic runs of a job.
rerun_max	Determines the maximum number of reruns that can be performed for the job. Integer.

Parameter	Description
rerun_member	(z/OS only)Defines the name of the JCL member to use when the job automatically reruns. String. Optional.
reten_days	Determines the number of days to retain the job in the History Jobs file. For z/OS jobs only.
reten_gen	Maximum number of generations of the job to keep in the History Jobs file. For z/OS jobs, only.
schedule_environment	(z/OS only)Indicates the JES2 workload management scheduling environment that is to be associated with the job. String.
shouts	Indicates a notification of a job's status. Seeshouts XML parameters (on page 125)
specific_times	Specific time for a cyclic job to run, such as 7:00 or 11:00. Limited to 4000 for all fields. This parameter is relevant only for jobs running in Control-M/Server version 6.4.01 and later and Control-M for z/OS version 7.0.01 and later.
step_ranges	Specifies a range of steps in the steps of an On PGMST statement. See step_range XML parameters (on page 124)
sys_db	(z/OS only) Indicates that a single data set is used for archiving the **SYSDATA** of all jobs until it is full, when another data set is started. Valid values: ▪ **yes:** Single data set created for the SYSDATA of each job run. ▪ **no:** Separate data set created for the SYSDATA of each job run.
output_from_class	(z/OS only) Limits the output handling operation to only outputs from the specified class.
output_option	Output Handling options. Optional.
output_parameter	Certain **output_option** values require that you supply additional information (such as **Copy**, **NewDest**): ▪ If the **output_option** element is **change_class**, the **output_parameter** value corresponds to the new class name. ▪ If the **output_option** element is copy, the **output_parameter** value corresponds to the destination file name. ▪ If the **output_option** element is move, the **output_parameter** value corresponds to the new destination for the file.

Parameter	Description
system_affinity	(z/Os jobs only) Indicates the identity of the system in which the job must be initiated and executed (in JES2). Also indicates the identity of the processor on which the job must execute (in JES3).
task_class	Control-D mission. Mandatory for Control-D jobs. Valid values: ▪ **distribution** ▪ **decollation**
task_type	Defines one or more parameters which determines what the job runs. **Microsoft Windows and UNIX:** ▪ job ▪ command ▪ dummy ▪ detached ▪ external ▪ SMART_folder ▪ sub_folder **Control-M for z/OS:** ▪ job ▪ task ▪ SMART_folder ▪ cyclic_job ▪ emergency_job ▪ emergency_cyclic_job ▪ cyclic_task ▪ emergency_task ▪ emergency_cyclic_task **NOTE:** SMART Folder: Specify **SMART_folder** as **task_type** parameter (not for a regular job).
time_due_out	Time that the job is expected to finish.
time_from	Indicates the earliest time for submitting the job.

Parameter	Description
time_reference	Valid values: ▪ **server** ▪ **adjust**
time_until	Enables the job to run the next day without time limitations.
time_until_days_offset	Defines the number of days after the original scheduling date of the job during which execution of the job can end. **NOTE:** This parameter is relevant only for jobs running in Control-M for z/OS version 6.2.00 and later.
time_zone	Indicates the time zone according to which the job should be scheduled.
tolerance	Maximum delay in minutes permitted for a late submission when selecting a specific time (e.g. 5 minutes).Maximum delay in minutes permitted for late submission when selecting a specific time. **NOTE:** This parameter is relevant only for jobs running in Control-M/Server version 6.4.01 and later and Control-M for z/OS version 7.0.01 and later.
variable_assignments	All variables are identified by the %% prefix. If %% is included in the value for a job processing parameter, Control-M assumes that it is referring to a variable or function. Variable_assignments include the following XML parameters: ▪ **name:** Defines the name of the variable ▪ **value:** Defines the value of the variable expression

on_statements XML parameters

The following table describes on_statements XML parameters:

Parameter	Sub parameter	Description
on_statement or		Specifies a sequence of **on_statements**, which consist of the following: ▪ **and_or** ▪ **code** ▪ **procedure_step** ▪ **program_step** ▪ **statement**
	and_or	Specifies the relationship between two successive items in a series. Optional.
	code	Defines the code value for the On Statement/Code parameter. Valid values: ▪ **ok** ▪ **not_ok**
	procedure_step	Defines a step in the procedure that triggers the On statement. String.
	program_step	Defines a step in the program that triggers the On statement. String.
	statement	Defines a **statement,** which can be either one of the following: ▪ A character string containing a statement from the job script file (1-132 characters). The specified string can be a portion of the statement. ▪ An asterisk (*****), when code is a completion status for a job.

Parameter	Sub parameter	Description
on_statement_output		Specifies a sequence of **on_statement_output,** which consists of the following: ▪ **find_output_pattern** ▪ **find_pattern_from** ▪ **find_pattern_to** ▪ **and_or**
	find_output_pattern	Defines a string of up to 40 characters.
	find_pattern_from	Refers to a number from 001 through 132, indicating the column at which the search should start. If this field is blank, the value 001 is assumed. The value in this field must be lower than that in the To Column field.
	find_pattern_to and_or	Refers to a number from 001 through 132, indicating the column at which the search should end. If this field is blank, the value 132 is assumed. The value in this field must be higher than that in the From Column field.
	and_or	Defines option buttons that set the logical relationship between multiple On statements.

do_statements type XML parameters

The following table describes do statements type XML parameters:

Parameter	Description
do	Refers to a sequence of **do_statements**.
do_variable	Assigns a variable when the On criteria is met. The parameter includes the following: ▪ **name**: Name of the item in question (for example, when specified for **request**, **name** is the name of the request; when specified for pipe, **name** is the name of the pipe) ▪ **value:** Value of the variable's expression.
do_cond	Assigns an In or Out condition when the On criteria are met. See do_cond XML parameters (on page 118)
do_ctbrule	Invokes a Control-M/Analyzer rule that executes during the processing of a specific program step when an On condition is met. The **do_ctbrule** XML parameters are as follows: ▪ **name**: Name of the Control-M/Analyzer rule ▪ **parameter**: Contains arguments that are passed to the Control-M/Analyzer rule
do_forcejob	Forces a specified job when the current job is performed. The parameter includes the following: ▪ **Control_m**: Defines the name of the target Control-M. Optional (string) ▪ **dsn**: Defines the name of the directory/library containing Table file. [z/OS only] ▪ **job**: Specifies the job name of the job that is forced ▪ **folder:** Defines the name of the folder with which the job specified in **do_forcejob** is associated ▪ **order_date:** Defines the original scheduling date of a job ▪ **variable_assignments**: Defines a sequence of variable_assignment XML parameters. Only relevant if Control_m has a value. Optional
do_ifrerun	Specifies job steps to be executed during rerun of a job. Only for networks using Control-M/Restart. See do_ifrerun XML parameters (on page 118)
do_mail	Sends e-mail. See do_mail XML parameters (on page 119)
do_remedy	Creates a remedy ticket. See do_remedy XML parameters (on page 120)

Parameter	Description
do_shout	Sends a shout message when the On criteria are met. See do_shout XML parameters (on page 120).
do_output	Determines what to do with the output documentation when On criteria are met. See do_output XML parameters (on page 121)

do_cond XML parameters

The following table describes do_cond XML parameters description:

Parameter	Description
condition	Defines a condition name. When specified, it is be accompanied by the other condition parameter element, **date** (and, optionally, by **sign** or **and_or**). It is wrapped in the **in_condition** and **out_condition** elements.
date	Specifies an order date for various condition formats.
sign	Indicates whether to add or delete an Out condition Valid values: ▪ **add** ▪ **delete**

do_ifrerun XML parameters

The following table describes do_ifrerun XML parameters:

Parameter	Description
confirm	Indicates that a job rerun specified by the Do If Rerun parameter must be manually confirmed before it is executed. Valid values: ▪ **yes:** Requires confirmation ▪ **no:** Confirmation is not required
from_procedure_step	Defines a procedure step (EXEC statement) that invokes a procedure from which the specified program step program is executed.

Parameter	Description
from_program_step	Defines a Job step. The execution results of the program executed by the job step are checked against the specified codes criteria.
to_procedure_step	Indicates a last procedure step in a range
to_program_step	Indicates a last program step in a range

do_mail XML parameters

The following table describes do_mail XML parameters:

Parameter	Description
cc	Optional additional address to which a Do Mail can be sent. Optional.
message	Text of the message. String.
to	Recipient of the **do_mail** message.
subject	Subject of the **do_mail** message.
urgency	Indicates the severity of a mail or shout message. Valid values: ▪ **regular** (Default) ▪ **urgent** ▪ **very_urgent**
attach_output	Specifies whether the output should be sent as an e-mail attachment. Valid values are: ▪ **yes -** Send the job's output as an attachment ▪ **no** - Do not send the job's output as an attachment ▪ **default -** Use the settings configured for the relevant Control-M server to determine whether the job's output should be sent as an attachment. **NOTE:** This parameter is relevant only for jobs running in Control-M/Server version 6.4.01 and later and Control-M for z/OS version 7.0.01 and later.

do_remedy XML parameters

The following table describes the do_remedy XML parameters:

Parameter	Description
urgency	Indicates the urgency of the Remedy ticket. Valid values are: ▪ low ▪ medium ▪ high ▪ urgent ▪ clear
description	Defines the description of the problem for which you are opening up the ticket.
summary	Defines a summary of the problem for which you are opening up a ticket.

do_shout XML parameters

The following table describes do_shout XML parameters:

Parameter	Description
destination	Indicates the recipient of a Shout message. Specified in both the Shout or the Notify parameters.
message	Defines the text of the message. String.
urgency	Indicates the severity of a mail or shout message. Valid values: ▪ **regular** (Default) ▪ **urgent** ▪ **very_urgent**

do_output XML parameters

The following table describes the do_output XML parameters:

Parameter	Description
from_class	Specifies the class of jobs with outputs that are handled using the Do Output specifications of the job.
option	Indicates the Do Output parameter output handling options. Valid values: ▪ **Release** ▪ **Delete** ▪ **Copy** ▪ **Move** ▪ **File** ▪ **NewDest (z/OS)** ▪ **ChangeClass (z/OS)** **NOTE:** Copy and Move are not used with z/OS.
parameter	Contains additional output handling information. The type of information required is dependent on the parameter value of the following **option** elements: ▪ ChangeClass: Enables the value corresponds to the new class name. ▪ Copy: Enables the value corresponds to the destination file name. ▪ Move: Enables the value corresponds to the new destination for the file.

ctb_step XML parameters

The following table describes the ctb_step XML parameters:

Parameter	Description
ctb_arguments	Defines the Control-M/Analyzer argument.
ctb_name	Defines the name of the Control-M/Analyzer entity. Must be a valid name of a Control-M/Analyzer rule or mission.
ctb_step_position	Indicates where to place the Control-M/Analyzer step in the job.
ctb_type	Defines the type of Control-M/Analyzer entity.

in_condition XML parameters

The following table describes in_condition XML parameters:

Parameter	Description
and_or	Specifies the relationship between two successive items in a series. Optional. Valid values: ▪ **and** ▪ **or**
condition	Defines the condition name.
date	Specifies an order date for various condition formats.

control_resources XML parameters

The following table describes control_resources XML parameters:

Parameter	Description
resource	Defines the name of the specified resource.
type	Indicates job access to a Control resource. Valid values are: ▪ **exclusive** - default ▪ **shared**
on_fail	Indicates whether to keep a Control resource tied to a job if the job does not end OK. Valid values: ▪ **keep** ▪ **release** - default

quantitative_resource XML parameters

The following table describes the quantitative_resource XML parameters:

Parameter	Description
quantity	Determines the amount of the specified quantitative resource.
resource	Defines the name of the specified resource.
on_ok	Indicates whether to keep a Quantitative resource tied to a job if the job ends OK. Valid values are: ▪ **release** ▪ **discard** **NOTE: on_ok** is relevant only for jobs running in Control-M for z/OS version 6.2.00 and later.
on_fail	Indicates whether to keep a Quantitative resource tied to a job if the job does not end OK. Valid values are: ▪ **keep** ▪ **release** **NOTE: on_fail** is relevant only for jobs running in Control-M for z/OS version 6.2.00 and later

pipe XML parameters

The following table describes pipe XML parameters:

Parameter	Description
flag	Valid values: ▪ yes ▪ no
name	Name of the item in question (for example, when specified for **request**, **name** is the name of the request; when specified for pipe, **name** is the name of the pipe)

out_condition XML parameters

The following table describes out_condition XML parameters:

Parameter	Description
condition	Condition name. When specified, it is be accompanied by the other condition parameter element, **date** (and, optionally, by **sign** or **and_or**).
date	Specifies an order date for various condition formats.
sign	Indicates whether to add or delete an Out condition Valid values: ▪ **add** ▪ **delete**

step_range XML parameters

The following table describes step_range XML parameters:

Parameter	Description
name	Name of the item in question
from_procedure_ step	Procedure step (EXEC statement) that invokes a procedure from which the specified program step program is executed
from_program _step	Job step. The execution results of the program executed by the job step are checked against the specified codes criteria
to_procedure _step	Last procedure step in a range
to_program_step	Last program step in a range

shouts XML parameters

The following table describes shouts XML parameters:

Parameter	Description
destination	Recipient of a Shout message. Specified in both the Shout or the Do Shout parameters.
message	Text of the message. String.
time	Time that the message is sent.
urgency	Indicates the severity of a mail or shout message. Valid values: ▪ regular (Default) ▪ urgent ▪ very_urgent
when	Time that the Shout message was sent. Valid values: ▪ ok ▪ not_ok ▪ rerun (not valid for SMART Folder entities) ▪ late_submission ▪ late_time ▪ execution_time
shout_days_offset	The number of days relative to the ODAT by which the sending of the Shout message is offset. Valid values are: ▪ a number from 0 through 254 ▪ blank – no offset **NOTE: shout_days_offset** is relevant only for jobs running in Control-M for z/OS version 6.2.00 and later.

interval_sequence XML parameters

The following table describes Interval_sequence XML parameters:

Parameter	Description
interval_item	Time interval to rerun a cyclic job such as +2H, +1D, or +30M. Limited to 4000 characters for all fields.

specific_times XML parameters

The following table describes the specific_times XML parameters:

Parameter	Description
specific_time	Specific time for a cyclic job to run, such as 7:00 or 11:00. Limited to 4000 for all fields.

captures XML parameters

The following table describes the captures XML parameters:

Parameter	Description
search_string	Defines the string in the output file to begin the capture process. Validation: String
skip_lines	Defines the number of lines to skip from the search string in the job output file. Minimum: 0
skip_columns	Defines the number of words or characters to skip in the job output file. Default value: 0
capture_by	Defines whether to capture by word or character. Default value: character

Parameter	Description
delimiter	Defines the character type that marks the beginning/end of the string. Valid values: ▪ White space ▪ space ▪ tab
count_to_take	Defines the number of words or characters to capture. Valid values: ▪ <number of words>/<number of characters> ▪ 0 - indicates until end of line
variable	Defines the variable type: Valid values: ▪ local variable ▪ global variable ▪ named pool variable ▪ SMART folder variable

response _create_aj XML parameters

The following table describes response_create_aj XML parameters:

Parameter	Description
status	Description of message content. String.
response_token	Used in the polling request.

For an example of a response, see Job creation examples (on page 129). Job creation errors are described in Create active job request errors (Major code 409) (on page 207).

NOTE: XML parameters for fault_create_aj and fault_poll_create_aj, as well as a sample fault response are described in Fault Response (on page 185).

request_poll_create_aj XML parameters

The following table describes request_poll_create_aj XML parameters:

Parameter	Description
user_token	Serial identification number supplied to the user during registration. String.
response_token	Used in a polling request. This token is received in the immediate response of a response_create_aj.

For an example of a poll request, see Job creation examples (on page 129).

response_poll_create_aj XML parameters

The following table describes response_poll_create_aj XML parameters:

Parameter	Description
status	Describes the condition of the element that contains it.(for example, **Error**). String.
jobs	A sequence of a **Job** parameter that indicates a single job, which includes the following parameters: ▪ **status:** Defines the condition of the element that contains it (such as Error). String ▪ job_data: **Job_data:** An element that contains other parameters that describe the job. A sequence of **job_data,** which contains the following parameters: ▪ **rba:** Relative block address. String ▪ **order_id:** Serial number assigned to the job by Control-M Workload Automation installation. String. ▪ **file_name:** Name of the file that contains the job script. String. ▪ **job_name:** Name of the job. String. ▪ **is_folder:** Indicates whether the job is a member of a SMART Folder. Valid values: **no** (not a member of a SMART Folder) or **yes** (member of a SMART Folder) ▪ **ret_text:** Text describing the job run. String.

For an example of a poll response, see Job creation examples (on page 129).

Job creation examples

The following topic describes examples of Job Creation request and responses together with examples to create cyclic jobs, jobs that include on do statements and jobs that create an active SMART folder:

request_create_aj XML example

The following example describes a successful request:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <ctmem:request_create_aj
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
            <ctmem:user_token>12345630</ctmem:user_token>
            <ctmem:control_m>ctm900</ctmem:control_m>
            <ctmem:active_job>
                <ctmem:job_name>MYJOB</ctmem:job_name>
                <ctmem:run_as>controlm</ctmem:run_as>
                <ctmem:task_type>command</ctmem:task_type>
                <ctmem:application>MYAPP</ctmem:application>
                <ctmem:sub_application>MYGROUP1</ctmem:sub_application>
                <ctmem:command>ls</ctmem:command>
            </ctmem:active_job>
        </ctmem:request_create_aj>
    </SOAP-ENV:Body>
```

```
</SOAP-ENV:Envelope>
```

response_create_aj XML example

The following example describes a successful response:

```
<?xml version="1.0" encoding="iso-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

    <SOAP-ENV:Body>

        <ctmem:response_create_aj
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

            <ctmem:status>OK</ctmem:status>

            <ctmem:response_token>96</ctmem:response_token>

        </ctmem:response_create_aj>

    </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

request_poll_create_aj example

The following example describes a successful request:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

    <SOAP-ENV:Body>

        <ctmem:request_poll_create_aj
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

            <ctmem:user_token>12345630</ctmem:user_token>

            <ctmem:response_token>96</ctmem:response_token>

        </ctmem:request_poll_create_aj>

    </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

response_poll_create_aj XML example

The following example describes a successful request:

```
<?xml version="1.0" encoding="iso-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

    <SOAP-ENV:Body>
```

```
        <ctmem:response_poll_create_aj
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
            <ctmem:status>OK</ctmem:status>
            <ctmem:jobs>
                <ctmem:job>
                    <ctmem:status>OK</ctmem:status>
                    <ctmem:job_data>
                        <ctmem:rba>000000</ctmem:rba>
                        <ctmem:order_id>0023e</ctmem:order_id>
                        <ctmem:file_name>
                        </ctmem:mem_name>
                        <ctmem:job_name>MYJOB</ctmem:job_name>
                        <ctmem:is_folder>no</ctmem:is_folder>
                    </ctmem:job_data>
                </ctmem:job>
            </ctmem:jobs>
        </ctmem:response_poll_create_aj>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Control and quantitative resources example

The following example describes a successful job creation request that creates a single job requiring Control and Quantitative resources.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <ctmem:request_create_aj
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
      <ctmem:user_token>$USER_TOKEN$</ctmem:user_token>
      <ctmem:control_m>ctm3-omega</ctmem:control_m>
      <ctmem:active_job>
        <ctmem:job_name>Rsc0</ctmem:job_name>
        <ctmem:mem_name>Rsc0</ctmem:mem_name>
        <ctmem:task_type>dummy</ctmem:task_type>
        <ctmem:application>Resource</ctmem:application>
```

```
<ctmem:sub_application>Resource</ctmem:sub_application>
<ctmem:max_wait>0</ctmem:max_wait>
<ctmem:description>Job requiring resources.</ctmem:description>
<ctmem:cyclic>no</ctmem:cyclic>
<ctmem:confirm_flag>yes</ctmem:confirm_flag>
<ctmem:sys_db>no</ctmem:sys_db>
<ctmem:arch_max_days>0</ctmem:arch_max_days>
<ctmem:arch_max_runs>0</ctmem:arch_max_runs>
<ctmem:override_path>asds</ctmem:override_path>
<ctmem:count_cyclic_from>end</ctmem:count_cyclic_from>
<ctmem:control_resources>
  <ctmem:control_resource>
    <ctmem:resource>Disks</ctmem:resource>
    <ctmem:type>shared</ctmem:type>
  </ctmem:control_resource>
  <ctmem:control_resource>
    <ctmem:resource>Time</ctmem:resource>
    <ctmem:type>shared</ctmem:type>
  </ctmem:control_resource>
  <ctmem:control_resource>
    <ctmem:resource>JJ4</ctmem:resource>
    <ctmem:type>shared</ctmem:type>
  </ctmem:control_resource>
  <ctmem:control_resource>
    <ctmem:resource>DE1a34</ctmem:resource>
    <ctmem:type>exclusive</ctmem:type>
  </ctmem:control_resource>
  <ctmem:control_resource>
    <ctmem:resource>CPU</ctmem:resource>
    <ctmem:type>shared</ctmem:type>
  </ctmem:control_resource>
</ctmem:control_resources>
<ctmem:quantitative_resources>
  <ctmem:quantitative_resource>
    <ctmem:resource>DL</ctmem:resource>
```

```
            <ctmem:quantity>30</ctmem:quantity>
          </ctmem:quantitative_resource>
        </ctmem:quantitative_resources>
      </ctmem:active_job>
    </ctmem:request_create_aj>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Create a job including in and out conditions example

The following example describes a successful Job Creation request that creates a single job including In and Out conditions.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <ctmem:request_create_aj
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
      <ctmem:user_token>$USER_TOKEN$</ctmem:user_token>
      <ctmem:control_m>ctm3-omega</ctmem:control_m>
      <ctmem:active_job>
        <ctmem:job_name>InCond0</ctmem:job_name>
        <ctmem:mem_name>InCond0</ctmem:mem_name>
        <ctmem:task_type>command</ctmem:task_type>
        <ctmem:application>InCond</ctmem:application>
        <ctmem:sub_application>InCond</ctmem:sub_application>
        <ctmem:max_wait>0</ctmem:max_wait>
        <ctmem:prevent_nct2>no</ctmem:prevent_nct2>
        <ctmem:time_from>0800</ctmem:time_from>
        <ctmem:critical>no</ctmem:critical>
        <ctmem:cyclic>no</ctmem:cyclic>
        <ctmem:confirm_flag>yes</ctmem:confirm_flag>
        <ctmem:auto_archive>no</ctmem:auto_archive>
        <ctmem:sys_db>no</ctmem:sys_db>
        <ctmem:arch_max_days>0</ctmem:arch_max_days>
        <ctmem:arch_max_runs>0</ctmem:arch_max_runs>
        <ctmem:rerun_max>0</ctmem:rerun_max>
```

```
        <ctmem:command>ls -l</ctmem:command>
        <ctmem:in_conditions>
          <ctmem:in_condition>
            <ctmem:condition>ctm600a0</ctmem:condition>
            <ctmem:date>ODAT</ctmem:date>
            <ctmem:and_or>and</ctmem:and_or>
          </ctmem:in_condition>
        </ctmem:in_conditions>
        <ctmem:out_conditions>
          <ctmem:out_condition>
            <ctmem:condition>ctm600a0</ctmem:condition>
            <ctmem:date>ODAT</ctmem:date>
            <ctmem:sign>delete</ctmem:sign>
          </ctmem:out_condition>
        </ctmem:out_conditions>
      </ctmem:active_job>
    </ctmem:request_create_aj>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Create a job requiring confirmation example

The following example describes a successful Job Creation request that creates a single job that performs a command and requires confirmation to run.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <ctmem:request_create_aj
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
      <ctmem:user_token>$USER_TOKEN$</ctmem:user_token>
      <ctmem:control_m>ctm3-omega</ctmem:control_m>
      <ctmem:active_job>
        <ctmem:job_name>Conf0</ctmem:job_name>
        <ctmem:mem_name>Conf0</ctmem:mem_name>
        <ctmem:task_type>command</ctmem:task_type>
        <ctmem:application>Confirm</ctmem:application>
```

```
        <ctmem:sub_application>Confirm</ctmem:sub_application>
        <ctmem:max_wait>0</ctmem:max_wait>
        <ctmem:prevent_nct2>no</ctmem:prevent_nct2>
        <ctmem:time_from>0900</ctmem:time_from>
        <ctmem:time_until>1100</ctmem:time_until>
        <ctmem:critical>no</ctmem:critical>
        <ctmem:cyclic>no</ctmem:cyclic>
        <ctmem:confirm_flag>yes</ctmem:confirm_flag>
        <ctmem:auto_archive>no</ctmem:auto_archive>
        <ctmem:sys_db>no</ctmem:sys_db>
        <ctmem:arch_max_days>0</ctmem:arch_max_days>
        <ctmem:arch_max_runs>0</ctmem:arch_max_runs>
        <ctmem:rerun_max>0</ctmem:rerun_max>
        <ctmem:command>ls -l</ctmem:command>
      </ctmem:active_job>
    </ctmem:request_create_aj>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Create a cyclic Job example

The following example describes a successful Job Creation request that creates a single cyclic job that performs a command.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <ctmem:request_create_aj
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
      <ctmem:user_token>$USER_TOKEN$</ctmem:user_token>
      <ctmem:control_m>ctm3-omega</ctmem:control_m>
      <ctmem:active_job>
        <ctmem:job_name>Cyc0</ctmem:job_name>
        <ctmem:mem_name>Cyc0</ctmem:mem_name>
        <ctmem:task_type>command</ctmem:task_type>
        <ctmem:application>Cyclic</ctmem:application>
        <ctmem:sub_application>Cyclic</ctmem:sub_application>
```

```
        <ctmem:max_wait>0</ctmem:max_wait>
        <ctmem:prevent_nct2>no</ctmem:prevent_nct2>
        <ctmem:time_from>1100</ctmem:time_from>
        <ctmem:time_until>2300</ctmem:time_until>
        <ctmem:rerun_interval>00060M</ctmem:rerun_interval>
        <ctmem:critical>no</ctmem:critical>
        <ctmem:cyclic>yes</ctmem:cyclic>
        <ctmem:confirm_flag>yes</ctmem:confirm_flag>
        <ctmem:auto_archive>no</ctmem:auto_archive>
        <ctmem:sys_db>no</ctmem:sys_db>
        <ctmem:arch_max_days>0</ctmem:arch_max_days>
        <ctmem:arch_max_runs>0</ctmem:arch_max_runs>
        <ctmem:rerun_max>0</ctmem:rerun_max>
        <ctmem:command>ls -l</ctmem:command>
      </ctmem:active_job>
    </ctmem:request_create_aj>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Create a job that includes On-Do statements (with variable) example

The following example describes a successful Job Creation request that creates a single job including multiple On-Do statements and a variable.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <ctmem:request_create_aj
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
      <ctmem:user_token>$USER_TOKEN$</ctmem:user_token>
      <ctmem:control_m>ctm3-omega</ctmem:control_m>
      <ctmem:active_job>
        <ctmem:job_name>DoSys0</ctmem:job_name>
        <ctmem:mem_name>DoSys0</ctmem:mem_name>
        <ctmem:task_type>command</ctmem:task_type>
        <ctmem:application>OnDo</ctmem:application>
        <ctmem:sub_application>DoSys</ctmem:sub_application>
```

```
        <ctmem:max_wait>0</ctmem:max_wait>
        <ctmem:description>Job with On-Do statements.</ctmem:description>
        <ctmem:time_from>0900</ctmem:time_from>
        <ctmem:time_until>1100</ctmem:time_until>
<ctmem:priority>Ab</ctmem:priority>
        <ctmem:critical>no</ctmem:critical>
        <ctmem:cyclic>no</ctmem:cyclic>
        <ctmem:confirm_flag>yes</ctmem:confirm_flag>
        <ctmem:auto_archive>no</ctmem:auto_archive>
        <ctmem:sys_db>no</ctmem:sys_db>
        <ctmem:arch_max_days>0</ctmem:arch_max_days>
        <ctmem:arch_max_runs>0</ctmem:arch_max_runs>
        <ctmem:rerun_max>0</ctmem:rerun_max>
        <ctmem:count_cyclic_from>end</ctmem:count_cyclic_from>
        <ctmem:command>ls -l</ctmem:command>
        <ctmem:variable_assignments>
          <ctmem:autoedit_assignment>
            <ctmem:name>COPYTO</ctmem:name>
            <ctmem:value>%%HOME.%%FILE</ctmem:value>
          </ctmem:autoedit_assignment>
        </ctmem:autoedit_assignments>
        <ctmem:on_do_statements>
        <ctmem:on_do_statement>
               <ctmem:on_statements>
                   <ctmem:on_statement>
                      <ctmem:code>*</ctmem:code>
                      <ctmem:statement>*</ctmem:statement>
                    </ctmem:on_statement>
               </ctmem:on_statements>
               <ctmem:do_statements>
                      <ctmem:do_output>
                             <ctmem:option>copy</ctmem:option>
                             <ctmem:parameter>%%COPYTO</ctmem:parameter>
                      </ctmem:do_output>
              </ctmem:do_statements>
```

```
            </ctmem:on_do_statement>
          </ctmem:on_do_statements>
        </ctmem:active_job>
      </ctmem:request_create_aj>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Create a job that includes On-Do statements example

The following example describes a successful Job Creation request that creates a single job including multiple On-Do statements.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
     <SOAP-ENV:Body>
          <ctmem:request_create_aj
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
               <ctmem:user_token>$USER_TOKEN$</ctmem:user_token>
               <ctmem:control_m>ctm3-omega</ctmem:control_m>
               <ctmem:active_job>
                    <ctmem:job_name>DoVAr0</ctmem:job_name>
                    <ctmem:mem_name>DoVAr0</ctmem:mem_name>
                    <ctmem:task_type>command</ctmem:task_type>
                    <ctmem:application>OnDo</ctmem:application>

<ctmem:sub_application>DoSetVar</ctmem:sub_application>
                    <ctmem:max_wait>0</ctmem:max_wait>
                    <ctmem:description>Multiple On-Dos</ctmem:description>
                    <ctmem:priority>Ab</ctmem:priority>
                    <ctmem:critical>no</ctmem:critical>
                    <ctmem:cyclic>no</ctmem:cyclic>
                      <ctmem:confirm_flag>no</ctmem:confirm_flag>
                    <ctmem:auto_archive>no</ctmem:auto_archive>
                    <ctmem:sys_db>no</ctmem:sys_db>
                    <ctmem:arch_max_days>0</ctmem:arch_max_days>
                    <ctmem:arch_max_runs>0</ctmem:arch_max_runs>
                    <ctmem:rerun_max>0</ctmem:rerun_max>
```

```
                    <ctmem:override_path>JOBDOC</ctmem:override_path>
                    <ctmem:count_cyclic_from>end</ctmem:count_cyclic_from>
                    <ctmem:command>echo %%ABC</ctmem:command>
                       <ctmem:on_do_statements>
                         <ctmem:on_do_statement>
                              <ctmem:on_statements>
                                   <ctmem:on_statement>
                                        <ctmem:code>*</ctmem:code>

<ctmem:statement>*</ctmem:statement>
                                   </ctmem:on_statement>
                              </ctmem:on_statements>
                              <ctmem:do_statements>
                                   <ctmem:do_variable>
                                        <ctmem:name>Auto1</ctmem:name>

<ctmem:value>1234567890</ctmem:value>
                                   </ctmem:do_variable
                              </ctmem:do_statements>
                         </ctmem:on_do_statement>
                    </ctmem:on_do_statements>
               </ctmem:active_job>
          </ctmem:request_create_aj>
     </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Create an active SMART folder example

The following example describes a successful request to create an active SMART folder:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <ctmem:request_create_aj
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
            <ctmem:user_token>$USER_TOKEN$</ctmem:user_token>
            <ctmem:control_m>omega</ctmem:control_m>
```

```
            <ctmem:active_job>
                <ctmem:task_type>SMART_folder</ctmem:task_type>
                <ctmem:application>appl</ctmem:application>
                <ctmem:sub_application>grp1</ctmem:sub_application>
                <ctmem:odate>ODAT</ctmem:odate>
            </ctmem:active_job>
        </ctmem:request_create_aj>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Create an active job in an existing SMART folder example

The following example describes a successful request to Create an active job in an existing SMART Folder:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <ctmem:request_create_aj
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
            <ctmem:user_token>$USER_TOKEN$</ctmem:user_token>
            <ctmem:control_m>omega-ctm3</ctmem:control_m>
            <ctmem:active_job>
                <ctmem:job_name>pwJob3</ctmem:job_name>
                <ctmem:mem_name>ajJob3</ctmem:mem_name>
                <ctmem:run_as>emuser</ctmem:run_as>
                <ctmem:task_type>command</ctmem:task_type>
                <ctmem:application>UnixAppl</ctmem:application>
                <ctmem:sub_application>UnixJobs</ctmem:sub_application>
                <ctmem:odate>ODAT</ctmem:odate>
                <ctmem:folder_id>0001l5</ctmem:folder_id>
                <ctmem:command>c:</ctmem:command>
            </ctmem:active_job>
        </ctmem:request_create_aj>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

NOTE: In the group ID element, you need to specify:

- The RBA of the SMART Folder if the Control-M is running on *z/OS*.
- The Order ID of the SMART Folder (with the addition of a leading zero) if the Control-M is running on another operating system.

Add condition

Adds prerequisite conditions. The following topics describe the add condition request parameters, polling request parameters, the response and polling response from Control-M/EM, together with examples:

request_add_condition XML parameters

The following table describes the request _add_condition XML parameters:

Parameter	Description
user_token	Serial identification number supplied to the user during registration. String.
control_m	Name of the Control-M installation that processes the request. String.
condition	Condition description wrapper. Wrapper for the **name** and **odate** elements that identify the specific condition being added or deleted (condition), which includes the following parameters: ▪ **name**: Name of the condition to add or delete ▪ **odate**: Order date of the condition. String. Valid values: mmdd and STAT.

For an example of a request, see request_add_condition example (on page 142).

request_add_condition example

The following example describes a successful request to adds the MYJOBOUTCOND1 prerequisite condition to the ctm900 Control-M installation:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

    <SOAP-ENV:Body>

        <ctmem:request_add_condition
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

            <ctmem:user_token>12345630</ctmem:user_token>

            <ctmem:control_m>ctm900</ctmem:control_m>

            <ctmem:condition>

                <ctmem:name>MYJOBOUTCOND1</ctmem:name>

                <ctmem:odate>STAT</ctmem:odate>

            </ctmem:condition>

        </ctmem:request_add_condition>

    </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

response_add_condition XML parameters

The following table describes response_add_condition XML parameters:

Parameter	Description
status	Description of message content. String.
response_token	Used in the polling request.

For an example of a response: response_add_condition XML example (on page 142). For error codes: See Add or Delete Condition request errors (Major code 404) (on page 204).

NOTE: XML parameters for fault_add_condition and fault_poll_add_condition, as well as a sample fault response are described in Fault Response (on page 185)

response_add_condition XML example

The following example describes a successful response to add a condition:

```
<?xml version="1.0" encoding="iso-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
```

```
    <SOAP-ENV:Body>
        <ctmem:response_add_condition
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
            <ctmem:status>OK</ctmem:status>
            <ctmem:response_token>97</ctmem:response_token>
        </ctmem:response_add_condition>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

request_poll_add_condition XML parameters

The following table describes request_poll_add_condition XML parameters:

Parameter	Description
user_token	Serial identification number supplied to the user during registration. String.
response_token	Used in a polling request. This token is received in the immediate response of a response_add_condition.

For an example of a polling request, see request_poll_add_condition XML example (on page 143).

request_poll_add_condition XML example

The following example describes a successful request:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <ctmem:request_poll_add_condition
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
            <ctmem:user_token>12345630</ctmem:user_token>
            <ctmem:response_token>97</ctmem:response_token>
        </ctmem:request_poll_add_condition>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

response_poll_add_condition XML parameters

The following table describes response_poll_add_condition XML parameters:

Parameter	Description
status	Description of message content (**OK** or **EXEC**). String.
response_data	Refers to **status**: Description of message content. String

For an example of a response, see response_poll_add_condition XML example (on page 144).

response_poll_add_condition XML example

The following example describes a successful response:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <ctmem:response_poll_add_condition
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
            <ctmem:status>OK</ctmem:status>
        </ctmem:response_poll_add_condition>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Delete condition

Deletes prerequisite conditions. The following topics describe the delete condition request parameters, polling request parameters, the response and polling response from Control-M/EM, together with examples:

request_delete_condition XML parameters

The following table describes request_delete_condition XML parameters:

Parameter	Description
control_m	Name of the Control-M installation that processes the request. String.
condition	Condition description wrapper, which includes the folllowing parameters: ▪ **name**: Name of the condition to be added or deleted. String. ▪ **order_date:** Order date of the condition. String. Valid values: **mmdd or STAT**

For an example of a request, see request_delete_condition XML example (on page 145).

request_delete_condition XML example

The following example describes a successful request to remove the MYJOBOUTCOND1 prerequisite condition from the ctm900 Control-M installation:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

    <SOAP-ENV:Body>

        <ctmem:request_delete_condition
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

            <ctmem:user_token>12345630</ctmem:user_token>

            <ctmem:control_m>ctm900</ctmem:control_m>

            <ctmem:condition>

                <ctmem:name>MYJOBOUTCOND1</ctmem:name>

                <ctmem:odate>STAT</ctmem:odate>

            </ctmem:condition>

        </ctmem:request_delete_condition>

    </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

response_delete_condition XML parameters

The following table describes response_delete_condition XML parameters:

Parameter	Description
status	Description of message content. String.
response_token	Used in the polling request.

For an example of a response, see response_delete_condition XML example (on page 146). For error codes: See Add or Delete Condition request errors (Major code 404) (on page 204).

NOTE: XML parameters for fault_delete_condition and fault_poll_delete_condition, as well as a sample fault response are described in Fault Response (on page 185).

response_delete_condition XML example

The following example describes a successful response to delete a condition:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <ctmem:response_delete_condition
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
            <ctmem:status>OK</ctmem:status>
            <ctmem:response_token>99</ctmem:response_token>
        </ctmem:response_delete_condition>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

request_poll_delete_condition XML parameters

The following table describes request_poll_delete_condition XML parameters:

Parameter	Description
user_token	Serial identification number supplied to the user during registration. String.
response_token	Used in a polling request. This token is received in the immediate response of a response_delete_condition.

For an example of a polling request, see request_poll_delete_condition XML parameter example (on page 147).

request_poll_delete_condition XML parameter example

The following example describes a successful request:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

    <SOAP-ENV:Body>

        <ctmem:request_poll_delete_condition
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

            <ctmem:user_token>12345630</ctmem:user_token>

            <ctmem:response_token>99</ctmem:response_token>

        </ctmem:request_poll_delete_condition>

    </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

response_poll_delete_condition XML parameters

The following table response_poll_delete_condition XML parameters:

Parameter	Description
status	Description of message content (**OK** or **EXEC**). String.
response_data	Includes status: Description of message content. String

For an example of a polling response, see response_poll_delete_condition XML example (on page 147).

response_poll_delete_condition XML example

The following example describes a successful response:

```
<?xml version="1.0" encoding="iso-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

    <SOAP-ENV:Body>

        <ctmem:response_poll_delete_condition
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

            <ctmem:status>OK</ctmem:status>

        </ctmem:response_poll_delete_condition>

    </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

List conditions

Lists prerequisite conditions.

The following topics describe the list condition request parameters, and the response from Control-M/EM, together with examples:

- request_list_conditions XML parameters (on page 148)
- response_list_conditions XML parameters (on page 150)
- request_list_conditions XML parameter example (on page 148)
- response_list_conditions XML parameter example (on page 150)

request_list_conditions XML parameters

The following table describes request_list_conditions XML parameters:

Parameter	Description
user_token	Defines the serial identification number supplied to the user during registration. String.
control_m	Defines the name of the Control-M installation that processes the request.
max_returned_nodes	Enables you to limit the number of returned entities (optional).
conditions_criterion	Defines the jobs criteria wrapper. String. Consists of the following filters: ▪ **include:** (Mandatory): Include filter definitions criteria wrapper ▪ **exclude**: (Optional): Exclude filter definitions criteria wrapper. These parameters include the **search_criterion wrapper** that consists of a sequence of param elements. At least one search_criterion element must appear in the include element. The amount of search_criterion elements is unlimited. The relationship between search_criterion elements in one filter is OR. For a description of param, see Search_criterion XML parameter (on page 164) table.

For an example of a request, see request_list_conditions XML parameter example (on page 148).

request_list_conditions XML parameter example

The following example describes a successful request to list a condition:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

 <SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

 <SOAP-ENV:Body>
```

```
 <ctmem:request_list_conditions
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
 <ctmem:user_token>$USER_TOKEN$</ctmem:user_token>
 <ctmem:control_m>$CONTROL_M$</ctmem:control_m>
 <ctmem:conditions_criterion>
<ctmem:include>
 <ctmem:search_criterion>
 <ctmem:param>
  <ctmem:name>CONDITION</ctmem:name>
  <ctmem:operator>LIKE</ctmem:operator>
  <ctmem:value>cond*</ctmem:value>
  </ctmem:param>
 <ctmem:param>
  <ctmem:name>ODATE</ctmem:name>
  <ctmem:operator>EQ</ctmem:operator>
  <ctmem:value>0207</ctmem:value>
  </ctmem:param>
  </ctmem:search_criterion>
  </ctmem:include>
  </ctmem:conditions_criterion>
  </ctmem:request_list_conditions>
  </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

response_list_conditions XML parameters

The following table describes request_delete_condition XML parameters:

Parameter	Description
list_conditions	Indicates a sequence of **list_condition_type**, which consists of the following parameters: ▪ **condition_name:** Indicates the name of the condition ▪ **control_m:** Indicates the Name of the Control-M installation that processes the request. String. ▪ **odate:** Indicates the order date of the condition. String. Valid values: mmdd and STAT.

For an example of a response, see response_list_conditions XML parameter example (on page 150).

response_list_conditions XML parameter example

The following example describes a successful response to list a condition:

```
<?xml version="1.0" encoding="iso-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

                <SOAP-ENV:Body>

                                    <ctmem:response_list_conditions
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

                                                        <ctmem:list_conditions>


<ctmem:condition_name>cond01

</ctmem:condition_name>

<ctmem:control_m>BMC-JTVY85J

</ctmem:control_m>

<ctmem:odate>0207

</ctmem:odate>

                                                        </ctmem:list_conditions>

                                                        <ctmem:list_conditions>


<ctmem:condition_name>cond02
```

```
</ctmem:condition_name>

<ctmem:control_m>BMC-JTVY85J

</ctmem:control_m>

<ctmem:odate>0207

</ctmem:odate>
                                                    </ctmem:list_conditions>
                                  </ctmem:response_list_conditions>
                  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Job actions in active jobs

Performs actions on jobs that are currently in the active jobs database. This request type allows the following job actions in active jobs for a single job:

- Hold: Enables you to stop a job from Control-M processing and allows you to update a job.
- Free: Enables you to free a previously held job.
- Confirm: Enables you to confirm submission of a job.
- Rerun: Enables you to rerun all jobs in the displayed list.
- Kill: Enables you to terminate a job while running in the middle of the execution.
- Set to OK: Enables you to set the status of a job to OK.

These requests are asynchronous. The immediate response indicates if the action request is sent (or not sent to Control-M). In case the request is successfully sent to Control-M, the response contains a response token. The returned response token can be used to get the final response using the existing polling request.

The following topics describe the job actions request parameters, polling request parameters, the response and polling response from Control-M/EM, together with examples:

Request XML parameters in active jobs database

The following table describes request XML parameters for job actions in active jobs database:

Name	Parameter name	Sub parameter
Hold	request_aj_hold	The following describes the sub parameters for each parameter: ▪ **user_token:** Serial identification number supplied to the user during registration. String. ▪ **control_m:** Control-M installation to which the job belongs. String. Mandatory. ▪ **order_id:** Order ID of the job. String. Mandatory.
Free	request_aj_free	
Confirm	request_aj_confirm	
Rerun	request_aj_rerun	
Kill	request_aj_kill	
Set to Ok	request_aj_set_to_ok	

For an example of a request, see request_aj_free xml parameter example (on page 152).

request_aj_free xml parameter example

The following example describes a successful request:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
<ctmem:request_aj_free xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
   <ctmem:user_token>12345630</ctmem:user_token>
   <ctmem:control_m>ctm900</ctmem:control_m>
   <ctmem:order_id>0023e</ctmem:order_id>
```

```
</ctmem:request_aj_free>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response XML parameters for job actions in the active jobs database

The following table describes the response parameters for job actions in the active jobs database:

Name	Parameter name	Sub parameter
Hold	response_aj_hold	The following describes the sub parameters for each parameter: ▪ **status:** The status of the response ▪ **response_token:** Used in a polling request
Free	response_aj_free	
Confirm	response_aj_confirm	
Rereun	response_aj_rerun	
Kill	response_aj_kill	
Set to Ok	response_aj_set_to_ok	

For an example of a response, see response_aj_free XML example (on page 153). For Errors, see Job actions request errors (Major code 450) (on page 210).

NOTE: XML parameters for fault_aj_force_ok and fault_poll_aj_force_ok, as well as a sample fault response are described in Fault Response (on page 185).

response_aj_free XML example

The following example describes a successful response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
        <ctmem:response_aj_free
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
              <ctmem:status>OK</ctmem:status>
              <ctmem:response_token>103</ctmem:response_token>
        </ctmem:response_aj_free>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Request poll XML parameters in active jobs database

The following table describes request_poll_aj XML parameters for job actions in active jobs database:

Name	Parameter name	Sub parameter
Hold	request_poll_aj_hold	The following describes the sub parameters for each parameter: ▪ **user_token:** Serial identification number supplied to the user during registration. String. ▪ **response_token**: Limits the number of returned entries.
Free	request_poll_aj_free	
Confirm	request_poll_aj_confirm	
Rerun	request_poll_aj_rerun	
Kill	request_poll_aj_kill	
Set to Ok	request_poll_aj_set_to_ok	

For an example of a polling request, see Request_polling_aj_free XML example (on page 154).

Request_polling_aj_free XML example

The following is a successful request:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

<SOAP-ENV:Body>

<ctmem:request_polling_aj_free
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

   <ctmem:user_token>12345630</ctmem:user_token>

           <ctmem:response_token>103</ctmem:response_token>

</ctmem:request_polling_aj_free>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Response poll XML parameters in active jobs database

The following table describes polling response parameters for job actions in active jobs database:

Name	Parameter name	Sub parameter
Hold	response_poll_aj_free	The following describes the sub parameters for each parameter: ▪ **status:** Status of polling. String. Valid values: **OK** or **EXEC** ▪ **control_m:** Control-M installation to which the job belongs. String. Mandatory. ▪ **order_id**: Order ID of the job. String. Mandatory.
Free	response_poll_aj_free	
Confirm	response_poll_aj_confirm	
Rerun	response_poll_aj_rerun	
Kill	response_poll_aj_kill	
Set to Ok	response_poll_aj_set_to_ok	

For an example of a polling response, see response_polling_aj_free XML example (on page 155).

response_polling_aj_free XML example

The following example describes the successful response:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

<SOAP-ENV:Body>

     <ctmem:response_polling_aj_free
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

               <ctmem:status>OK</ctmem:status>

               <ctmem:control_m>ctm900</ctmem:control_m>

               <ctmem:order_id>0023e</ctmem:order_id>

     </ctmem:response_polling_aj_free>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Job tracking

Tracks the progress of existing jobs in the Control-M installation. The following topics describe the job tracking request parameters, and the response from Control-M/EM, together with examples:

request_job_track XML parameters

The following table describes request_job_track XML parameters:

Parameter	Description
user_token	Defines a serial identification number supplied to the user during registration. String.
jobs	Indicates a sequence of **job**, which includes the following XML parameters: ▪ **control_m:** Control-M installation to which the job belongs. String. Mandatory. ▪ **order_id:** Order ID of the job. String. Mandatory.

For an example of a request, see request_job_track XML parameters (tracking single job) example (on page 157) and request_job_track XML parameters (tracking multiple jobs) example (on page 156).

request_job_track XML parameters (tracking multiple jobs) example

The following example describes a successful request to track multiple jobs:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

    <SOAP-ENV:Body>

        <ctmem:request_job_track
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

            <ctmem:user_token>12345630</ctmem:user_token>

            <ctmem:jobs>
```

```
                <ctmem:job>
                    <ctmem:control_m>ctm900</ctmem:control_m>
                    <ctmem:order_id>013ic</ctmem:order_id>
                </ctmem:job>
                <ctmem:job>
                    <ctmem:control_m>ctm900</ctmem:control_m>
                    <ctmem:order_id>006b2</ctmem:order_id>
                </ctmem:job>
                <ctmem:job>
                    <ctmem:control_m>ctm900</ctmem:control_m>
                    <ctmem:order_id>013xc</ctmem:order_id>
                </ctmem:job>
            </ctmem:jobs>
        </ctmem:request_job_track>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

request_job_track XML parameters (tracking single job) example

The following example describes a successful request to track a single job:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <ctmem:request_job_track
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
            <ctmem:user_token>12345630</ctmem:user_token>
            <ctmem:jobs>
                <ctmem:job>
                    <ctmem:control_m>ctm900</ctmem:control_m>
                    <ctmem:order_id>013ic</ctmem:order_id>
                </ctmem:job>
            </ctmem:jobs>
        </ctmem:request_job_track>
    </SOAP-ENV:Body>
```

```
</SOAP-ENV:Envelope>
```

response_job_track XML parameters

The following table describes response_job_track XML parameters:

Parameter	Description
status	Describes the condition of the element that contains it. String. **NOTE: status** is a descriptive element in the **response** and **job** elements.
jobs	Indicates a sequence of **job**, which includes the following parameters: ▪ **status:** Describes the condition of the element that contains it (such as, **Error**). String. ▪ **job_data:** An element that contains other parameters that describe the job A sequence of **job_data**. See job_data XML parameters (on page 159). ▪ **error_list:** A sequence of **error**. See Fault Response (on page 185) ▪ **error_list attribute: highest_severity:** Indicates the severity level of the most critical error included in the error list. If only one error is included, the severity for that error is displayed. String.
error_list	Indicates a sequence of **error**. See Fault Response (on page 185)

For an example of a response, see response_job_track XML parameters (tracking single job) example (on page 161) and response_job_track XML parameters (tracking multiple jobs) example (on page 160). For error codes, see Job tracking request errors (Major code 406) (on page 206).

job_data XML parameters

The following table describes job_data XML parameters:

Parameter	Description
control_m	String.
order_id	Serial number assigned to the job by the Control-M installation. String.
job_status	Execution status of the job. String. Valid values are: ▪ Ended OK ▪ Ended not OK ▪ Executing ▪ Wait Condition ▪ Wait Resource ▪ Wait User ▪ Not in AJF ▪ Unknown
state_digits	Serial number identifying the current job state.
order_date	Original Scheduling date of the job. String.

fault_job_track XML parameters

The following table describes fault_job_track XML parameters:

Parameter	Description	
jobs	A sequence of job.	
error_list	A sequence of **error**. See Fault Response (on page 185).	
error_list attribute:	highest_severity	Indicates the severity level of the most critical error included in the error list. If only one error is included, the severity for that error is displayed. String.

response_job_track XML parameters (tracking multiple jobs) example

The following example describes a successful response to track multiple jobs:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <ctmem:response_job_track
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
            <ctmem:status>OK</ctmem:status>
            <ctmem:jobs>
                <ctmem:job>
                    <ctmem:status>OK</ctmem:status>
                    <ctmem:job_data>
                        <ctmem:control_m>ctm900</ctmem:control_m>
                        <ctmem:order_id>013ic</ctmem:order_id>
                        <ctmem:job_status>Ended OK</ctmem:job_status>

<ctmem:state_digits>000030000000</ctmem:state_digits>
                        <ctmem:odate>050711</ctmem:odate>
                    </ctmem:job_data>
                </ctmem:job>
                <ctmem:job>
                    <ctmem:status>OK</ctmem:status>
                    <ctmem:job_data>
                        <ctmem:control_m>ctm900</ctmem:control_m>
                        <ctmem:order_id>006b2</ctmem:order_id>
                        <ctmem:job_status>Wait Resource</ctmem:job_status>

<ctmem:state_digits>220000004000</ctmem:state_digits>
                        <ctmem:odate>050704</ctmem:odate>
                    </ctmem:job_data>
                </ctmem:job>
                <ctmem:job>
                    <ctmem:status>OK</ctmem:status>
```

```
                    <ctmem:job_data>
                        <ctmem:control_m>ctm900</ctmem:control_m>
                        <ctmem:order_id>012qh</ctmem:order_id>
                        <ctmem:job_status>Ended OK</ctmem:job_status>

<ctmem:state_digits>000030000000</ctmem:state_digits>
                        <ctmem:odate>050711</ctmem:odate>
                    </ctmem:job_data>
                </ctmem:job>
            </ctmem:jobs>
        </ctmem:response_job_track>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

response_job_track XML parameters (tracking single job) example

The following example describes a successful response to track a single job:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <ctmem:response_job_track
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
            <ctmem:status>OK</ctmem:status>
            <ctmem:jobs>
                <ctmem:job>
                    <ctmem:status>OK</ctmem:status>
                    <ctmem:job_data>
                        <ctmem:control_m>ctm900</ctmem:control_m>
                        <ctmem:order_id>013ic</ctmem:order_id>
                        <ctmem:job_status>Ended OK</ctmem:job_status>

<ctmem:state_digits>000030000000</ctmem:state_digits>
                        <ctmem:odate>050711</ctmem:odate>
                    </ctmem:job_data>
                </ctmem:job>
```

```
            </ctmem:jobs>
        </ctmem:response_job_track>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Retrieve jobs in the active jobs database

Retrieves jobs in the active jobs database. Jobs are searched in the "All Jobs" collection in GUI Server memory. If the **PinAllJobsCollection** system parameter is turned on, the "All Jobs" collection is loaded into memory at start up of the GUI Server. If the **PinAllJobsCollection** is turned off, the request causes the All Jobs collection to be loaded into GUI Server memory, which may affect performance and response time.

The **EMAPIActiveJobsLoadLimit** system parameter controls the number of jobs in the active jobs database that are checked by the GUI Server when processing the request_act_retrieve_jobs request, and included in the request response. It is possible to limit the number of returned jobs on the client side by specifying the max_returned_nodes parameter in the request. The value of the max_returned_nodes parameter should not exceed the value of the **EMAPIActiveJobsLoadLimit** system parameter.

If the value of the max_returned_nodes parameter exceeds the value of the **EMAPIActiveJobsLoadLimit** system parameter, a fault response is returned. For more information about the **PinAllJobsCollection** and **EMAPIActiveJobsLoadLimit** system parameters, see GUI Server parameters.

The following topics describe the retrieve jobs request parameters, and the response from Control-M/EM, together with examples:

- request_act_retrieve_jobs XML Parameters (on page 163)
- request_act_retrieve_jobs XML example (on page 168)
- response_act_retrieve_jobs XML parameters (on page 170)
- Response_act_retrieve_jobs XML example (on page 173)

request_act_retrieve_jobs XML Parameters

The following table describes request_act_retrieve_jobs XML Parameters:

Parameter	Description
user_token	Serial identification number supplied to the user during registration. String.
max_returned_nodes	Limits the number of returned entities. Optional. **NOTE:** Should not exceed the value of the EMAPIActiveJobsLoadLimit system parameter.
retrieve_jobs_criterion	Retrieve jobs criteria wrapper. String. Consists of the following filters: ▪ **include:** (Mandatory): Include filter definitions criteria wrapper ▪ **exclude**: (Optional): Exclude filter definitions criteria wrapper. These parameters include the **search_criterion wrapper** that consists of a sequence of param elements. At least one search_criterion element must appear in the include element. The amount of search_criterion elements is unlimited. The relationship between search_criterion elements in one filter is OR. For a description of param, see Search_criterion XML parameter (on page 164).

For an example of request, see request_act_retrieve_jobs XML example (on page 168).

Search_criterion XML parameter

The following table describes search_criterion XML parameter:

Parameter	Description
param	Search criteria parameter wrapper. String. Parameters used to build the search criteria. At least one param element should appear under a search_criterion element. The amount of param elements is unbounded. The relationship between param elements in the same search_criterion is AND. Includes the following parameters: ▪ **name:** Mandatory. Name of the retrieve active job parameter used as a search criteria. String. For a list of valid values for the name parameter, see Valid name parameter values (on page 165) ▪ **operator:** Operator used in search criteria. String. Valid values: ▪ EQ ▪ NE ▪ LT ▪ GT ▪ LIKE ▪ **value:** Value used in search criteria. Any valid value of a job parameter. Wildcards and search patterns can be used in combination with LIKE operator. String. Mandatory.

Valid name parameter values

The following table describes Valid name parameter values:

Parameter	Description
order_id	(Mandatory) Defines the Order ID of the job to be retrieved. String.
data_center	Defines the data center to which the job belongs
application	Defines the application name to which the job's group belongs
appl_type	Defines the external application on which the job runs
sub_application	Defines the group name to which the job belongs
MEMNAME	Indicates the name of the file that contains the job script, or for z/OS jobs, the name of a member that contains one of the following in relation to the job to be executed: ▪ The JCL of the job ▪ The started task procedure ▪ Warning messages
job_name	Defines the job name

Parameter	Description
task_type	Defines one or more parameters which determines what the job runs. Valid values: **Microsoft Windows and UNIX:** ▪ Job ▪ Command ▪ Dummy ▪ Detached ▪ External ▪ SMART Folder ▪ Sub-folder **Control-M for z/OS:** ▪ Job ▪ Started Task ▪ SMART Folder ▪ Cyclic Job ▪ Emergency Job ▪ Emergency Cyclic Job ▪ Cyclic Task ▪ Emergency Task ▪ Emergency Cyclic Task
critical	Determines whether the job is a critical-path job in Control-M, which ensures resources allocation order.
cyclic	Indicates that the job must run at a designated time, interval of time.If selected, indicates that the current job is cyclic (it should be rerun at specified intervals).
emergency	Determines whether the z/OS job is an Emergency job.
part_of_BIM_service	Indicates if the job is included in a Business Service.
status	Indicates the job execution status.

Parameter	Description
ended	Indicates that the job ended. Valid values are: ▪ True ▪ False
ended_not_ok	Indicates that the job ended unsuccessfully.Valid values are: ▪ True ▪ False
ended_ok	Indicates that the job ended successfully. Valid values are: ▪ True ▪ False
late	Indicates that the job ended late.Valid values are: ▪ True ▪ False
held	Indicates that the job was held. Valid values are: ▪ True ▪ False
delete_flag	Indicates that the job was deleted. Valid values are: ▪ True ▪ False
requested	Indicates that the job was requested. Valid values are: ▪ True ▪ False
run_as	Identifies the user name with the authorization to execute the job. This parameter is used by the Control-M security mechanism.
host_group	Defines the name of a Control-M/Agent computer, remote host computer, or host group where the job is submitted.
description	(From Forecast only) Provides a description of the job in free text. A well written description can help you determine why the job was defined and how it fits into your business workflow.
order_date	Defines the job order date

Parameter	Description
AVG_RUNTIME	Indicates the average runtime of the job
START_TIME	Refers to the start time of the job
END_TIME	Refers to the end time of the job
Incond Name	Defines the name of the in-condition
Incond Date	Defines the date of the in-condition
Outcond Name	Defines the name of the out-condition
Outcond Date	Defines the date of the out-condition
Quant Res	Defines the name of the quantitative resource
Control Res	Defines the name of the control resource
RBA	Defines the relative block address (RBA). String.

request_act_retrieve_jobs XML example

The following example describes a successful request to retrieve jobs in the active jobs database:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <ctmem:request_act_retrieve_jobs
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
            <ctmem:user_token>12345630</ctmem:user_token>
            <ctmem:max_returned_nodes>1000</ctmem:max_returned_nodes>
            <ctmem:retrieve_jobs_criterion>
                <ctmem:include>
                    <ctmem:search_criterion>
                        <ctmem:param>
                            <ctmem:name>JOB_NAME</ctmem:name>
                            <ctmem:operator>LIKE</ctmem:operator>
                            <ctmem:value>*</ctmem:value>
```

```
                        </ctmem:param>
                        <ctmem:param>
                            <ctmem:name>ODATE</ctmem:name>
                            <ctmem:operator>EQ</ctmem:operator>
                            <ctmem:value>060625</ctmem:value>
                        </ctmem:param>
                    </ctmem:search_criterion>
                </ctmem:include>
                <ctmem:exclude>
                    <ctmem:search_criterion>
                        <ctmem:param>
                            <ctmem:name>DATA_CENTER</ctmem:name>
                            <ctmem:operator>NE</ctmem:operator>
                            <ctmem:value>ctm630</ctmem:value>
                        </ctmem:param>
                    </ctmem:search_criterion>
                    <ctmem:search_criterion>
                        <ctmem:param>
                            <ctmem:name>APPLICATION</ctmem:name>
                            <ctmem:operator>EQ</ctmem:operator>
                            <ctmem:value>AJMN</ctmem:value>
                        </ctmem:param>
                    </ctmem:search_criterion>
                    <ctmem:search_criterion>
                        <ctmem:param>
                            <ctmem:name>APPLICATION</ctmem:name>
                            <ctmem:operator>EQ</ctmem:operator>
                            <ctmem:value>FullJobs</ctmem:value>
                        </ctmem:param>
                    </ctmem:search_criterion>
                </ctmem:exclude>
            </ctmem:retrieve_jobs_criterion>
        </ctmem:request_act_retrieve_jobs>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

response_act_retrieve_jobs XML parameters

The following table describes response_act_retrieve_jobs XML parameters:

Parameter	Description
retrieved_nodes_number	Indicates a number of retrieved nodes.
jobs	Indicates a sequence of **job_data**. For more information, refer to job_data XML Parameters Description (on page 170)

For an example of a response, see Response_act_retrieve_jobs XML example (on page 173).

For error codes, see Retrieve active jobs request errors (Major code 440) (on page 210).

NOTE: XML parameters for fault_act_retrieve_jobs, as well as a sample fault response are described in Fault Response (on page 185).

job_data XML Parameters Description

The following table describes job_data XML Parameters:

Parameter	Description
application	Provides a logical name for sorting groups of jobs. This parameter is used to supply a common descriptive name to a set of related job groups. The jobs do not necessarily have to run at the same time.
application_type	Indicates the type of external application (such as SAP) on which the external application job runs.
average_runtime	Average time (in minutes) for the job to run. The field displays the runtime statistic generated by the latest run of the CTMJSA utility (which generates a statistic based on the last successful runs of the job).
control_m	Control-M installation to which the job belongs. String. Mandatory.
critical	Determines whether the job is a critical-path job in Control-M, which ensures resources allocation order.
cyclic	Indicates that the job must run at a designated time, interval of time.
description	Defines the text description of the job.
emergency	(Control-M for z/OS only)Indicates that the current job or started task is an emergency job or started task
end_time	Date and time the job finished executing.

Parameter	Description
file_name	Defines the name of the folder. In the Properties pane this parameter indicates the folder where the job belongs.
file_path	For non-z/OS jobs, File Path indicates the location of the file that contains the script. For z/OS jobs, Member Library indicates the location of the Member that contains the JCL, started task procedure, or Warning message.
folder_rba	RBA of job's SMART Folder entity.
in_BIM_service	Indicates if the job is included in Business Service.
job_name	Defines the name of the job.
job_state	Indicates the job state, such as Held, Deleted, or Restarted.
job_status	Name of the job. String. Valid values are: ▪ Ended OK ▪ Ended not OK ▪ Executing ▪ Wait Condition ▪ Wait Resource ▪ Wait User ▪ Not in AJF ▪ Unknown
order_date	Original scheduling date of a job.
order_id	Order ID of the job. String. Mandatory.
order_folder	Default or dummy folder to which you indicate the job belongs. A folder is not necessary because jobs that are created with Control-M/EM Web Services API are inserted directly into Active Jobs database. However, you may want to include a value for this parameter so that the job can be tracked during statistical analysis that uses folder as a criterion.

Parameter	Description
order_library	Default or dummy folder library in which folder documentation is said to be stored. A folder (and, by extension, a folder library) are not necessary because jobs that are created with the Control-M/EM Web Services API are inserted directly into Active Jobs database. However, you may want to include a value for this parameter so that the job can be tracked during statistical analysis that uses **Folder** or **Folder Library** as criteria. This parameter is specified only for z/OS jobs for which the **order_folder** element was also specified.
otime	Order time of the job. String. This parameter only applies to jobs run on Control-M/Server for Distributed Systems 6.3.0*x* or Control-M for z/OS 6.2.*xx* and above.
next_time	Indicates the next expected submission time for the job. For reruns or cyclic jobs that use the Interval option. For Control-M for z/OS jobs, only the time can be specified. For all other jobs, the time and the date can be specified.
rba	Relative block address (RBA). String.
run_as	Identifies the user name with the authorization to execute the job. This parameter is used by the Control-M security mechanism.
rerun_counter	Number of times the job has been rerun.
start_time	Date and time the job began executing.
state_digits	Serial number identifying the current job state.
sub_application	Defines the name of the sub application to which the job belongs. Used as a descriptive name for related groups of jobs.

Parameter	Description
task_type	Defines the type of the job (task) to be performed by Control-M. Valid values: **Microsoft Windows and UNIX:** ▪ Job ▪ Command ▪ Dummy ▪ Detached ▪ External ▪ SMART Folder ▪ Sub-folder **Control-M for z/OS:** ▪ Job ▪ Started Task ▪ SMART Folder ▪ Cyclic Job ▪ Emergency Job ▪ Emergency Cyclic Job ▪ Cyclic Task ▪ Emergency Task ▪ Emergency Cyclic Task
time_from	Indicates the earliest time for submitting the job.
time_until	Indicates the latest time for submitting the job.
time_zone	Indicates the global time zone used to calculate the interval for time-related conditions.

Response_act_retrieve_jobs XML example

The following example describes a successful response to retrieve jobs in the active jobs database:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
```

```
        <ctmem:response_act_retrieve_jobs
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
            <ctmem:retrieved_nodes_number>3
            </ctmem:retrieved_nodes_number>
            <ctmem:jobs>
                <ctmem:job>
                    <ctmem:job_data>
                        <ctmem:control_m>ctm900</ctmem:control_m>
                        <ctmem:order_id>0023e</ctmem:order_id>
                        <ctmem:rba>00023e</ctmem:rba>
                        <ctmem:folder_rba>000000</ctmem:folder_rba>
                        <ctmem:application>MYAPP</ctmem:application>

<ctmem:application_type>OS</ctmem:application_type>

<ctmem:sub_application>MYGROUP1</ctmem:sub_application>
                        <ctmem:job_name>MYJOB</ctmem:job_name>
                        <ctmem:mem_name> </ctmem:mem_name>
                        <ctmem:mem_lib> </ctmem:mem_lib>
                        <ctmem:order_folder> </ctmem:order_folder>
                        <ctmem:run_as>controlm</ctmem:run_as>
                        <ctmem:description></ctmem:description>
                        <ctmem:task_type>Command</ctmem:task_type>
                        <ctmem:time_zone>  </ctmem:time_zone>
                        <ctmem:in_BIM_service>0</ctmem:in_BIM_service>
                        <ctmem:job_status>Wait Resource </ctmem:job_status>
                        <ctmem:job_state>   </ctmem:job_state>

<ctmem:state_digits>020000004000</ctmem:state_digits>
                        <ctmem:odate>060625</ctmem:odate>
                        <ctmem:otime>20060625115417</ctmem:otime>
                        <ctmem:next_time>    </ctmem:next_time>
                        <ctmem:rerun_counter>00000</ctmem:rerun_counter>

<ctmem:average_runtime>000000</ctmem:average_runtime>
                        <ctmem:start_time>     </ctmem:start_time>
```

```
                        <ctmem:end_time>      </ctmem:end_time>
                        <ctmem:critical>0</ctmem:critical>
                        <ctmem:cyclic>False</ctmem:cyclic>
                        <ctmem:emergency>False</ctmem:emergency>
                    </ctmem:job_data>
                </ctmem:job>
                <ctmem:job>
                    <ctmem:job_data>
                        <ctmem:control_m>ctm630</ctmem:control_m>
                        <ctmem:order_id>0023d</ctmem:order_id>
                        <ctmem:rba>00023d</ctmem:rba>
                        <ctmem:folder_rba>000000</ctmem:folder_rba>
                        <ctmem:application>apiWinApp2</ctmem:application>
                        <ctmem:application_type>
</ctmem:application_type>

<ctmem:sub_application>apiGroup2</ctmem:sub_application>
                        <ctmem:job_name>apitest2</ctmem:job_name>
                        <ctmem:mem_name>apiMemName2</ctmem:mem_name>

<ctmem:mem_lib>d:&lt;OWNER&gt;controlm</ctmem:mem_lib>
                        <ctmem:order_folder>apitest</ctmem:order_folder>
                        <ctmem:owner>     </ctmem:owner>
                        <ctmem:description>    </ctmem:description>
                        <ctmem:task_type>Job  </ctmem:task_type>
                        <ctmem:time_zone>   </ctmem:time_zone>
                        <ctmem:in_BIM_service>0 </ctmem:in_BIM_service>
                        <ctmem:job_status>Wait Resource </ctmem:job_status>
                        <ctmem:job_state>  </ctmem:job_state>

<ctmem:state_digits>020000004000</ctmem:state_digits>
                        <ctmem:odate>060625</ctmem:odate>
                        <ctmem:otime>20060625113440</ctmem:otime>
                        <ctmem:next_time>   </ctmem:next_time>
                        <ctmem:rerun_counter>00000</ctmem:rerun_counter>
```

```
<ctmem:average_runtime>000000</ctmem:average_runtime>
                        <ctmem:start_time>  </ctmem:start_time>
                        <ctmem:end_time> </ctmem:end_time>
                        <ctmem:critical>0</ctmem:critical>
                        <ctmem:cyclic>False</ctmem:cyclic>
                        <ctmem:emergency>False</ctmem:emergency>
                    </ctmem:job_data>
                </ctmem:job>
                <ctmem:job>
                    <ctmem:job_data>
                        <ctmem:control_m>ctm630</ctmem:control_m>
                        <ctmem:order_id>0023c</ctmem:order_id>
                        <ctmem:rba>00023c</ctmem:rba>
                        <ctmem:folder_rba>000000</ctmem:folder_rba>
                        <ctmem:application>apiWinApp1</ctmem:application>
                        <ctmem:application_type> </ctmem:application_type>
                        <ctmem:sub_application>apiGroup1
</ctmem:sub_application>
                        <ctmem:job_name>apitest1</ctmem:job_name>
                        <ctmem:mem_name>apiMemName1</ctmem:mem_name>
                        <ctmem:mem_lib>   </ctmem:mem_lib>
                        <ctmem:order_folder>apitest</ctmem:order_folder>
                        <ctmem:run_as>controlm</ctmem:run_as>
                        <ctmem:description>    </ctmem:description>
                        <ctmem:task_type>Dummy </ctmem:task_type>
                        <ctmem:time_zone>  </ctmem:time_zone>
                        <ctmem:in_BIM_service>0</ctmem:in_BIM_service>
                        <ctmem:job_status>Ended OK</ctmem:job_status>
                        <ctmem:job_state>Run (2)</ctmem:job_state>

<ctmem:state_digits>000030000000</ctmem:state_digits>
                        <ctmem:odate>060625</ctmem:odate>
                        <ctmem:otime>20060625113439</ctmem:otime>
                        <ctmem:next_time>   </ctmem:next_time>
                        <ctmem:rerun_counter>00001</ctmem:rerun_counter>
```

```
<ctmem:average_runtime>000000</ctmem:average_runtime>

<ctmem:start_time>20060625113441</ctmem:start_time>
                        <ctmem:end_time>20060625113441</ctmem:end_time>
                        <ctmem:critical>0</ctmem:critical>
                        <ctmem:cyclic>False</ctmem:cyclic>
                        <ctmem:emergency>False</ctmem:emergency>
                    </ctmem:job_data>
                </ctmem:job>
            </ctmem:jobs>
        </ctmem:response_act_retrieve_jobs>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Change alert status

Changes the status (for example, from **not_noticed** to **handled**) of an alert. The following topics describe the change alert status request parameters, and the response from Control-M/EM, together with examples:

- request_change_alert_status XML parameters (on page 178)
- request_change_alert_status XML parameters example (on page 178)
- response_change_alert_status XML parameters (on page 179)
- response_change_alert_status failure example (on page 179)

request_change_alert_status XML parameters

The following table describes request_change_alert_status XML parameters:

Parameter	Description
alert_id	The ID number of the alert. String. For more information on monitoring and handling alerts, see Monitoring.
status	Required status for the specified alert. Valid values: ▪ **notice** ▪ **unnotice** ▪ **handle** ▪ **unhandle**

For an example of a request, see request_change_alert_status XML parameters example (on page 178).

request_change_alert_status XML parameters example

The following example describes a successful request to change alert status:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

    <SOAP-ENV:Body>

        <ctmem:request_change_alert_status
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

            <ctmem:user_token>12345630</ctmem:user_token>

            <ctmem:alert_id>359</ctmem:alert_id>

            <ctmem:status>notice</ctmem:status>

        </ctmem:request_change_alert_status>

    </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

response_change_alert_status XML parameters

The following table describes response_change_alert_status XML parameters:

Parameter	Description
status	Description of message content (for example, **Error**). String.

For an example of a response, see Response_change_alert_status successful example (on page 180) and Response_change_alert_status successful example (on page 180).

error codes: See Alerts request errors (Major code 408) (on page 207).

NOTE: XML parameters for the fault_change_alert_status, as well as a sample fault response are described in Fault Response (on page 185).

response_change_alert_status failure example

The following example describes a failure response to change alert status:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <SOAP-ENV:Fault>
            <faultcode>SOAP:Server</faultcode>
            <faultstring>Error response from EM Server</faultstring>
            <detail>
                <ctmem:fault_change_alert_status
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
                    <ctmem:error_list ctmem:highest_severity="Error">
                        <ctmem:error ctmem:major="408" ctmem:minor="1"
ctmem:severity="Error">
                            <ctmem:error_message>
                                Alert id is not valid.
                            </ctmem:error_message>
                        </ctmem:error>
                    </ctmem:error_list>
                </ctmem:fault_change_alert_status>
            </detail>
        </SOAP-ENV:Fault>
    </SOAP-ENV:Body>
```

```
</SOAP-ENV:Envelope>
```

Response_change_alert_status successful example

The following example describes a successful response to change alert status:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

    <SOAP-ENV:Body>

        <ctmem:response_change_alert_status
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">

            <ctmem:status>OK</ctmem:status>

        </ctmem:response_change_alert_status>

    </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Retrieve BIM Services list

Retrieves the list of services active in the Batch Impact Manager Server. The following topics describe the retrieve BIM Services list request parameters, and response from Control-M/EM, together with examples:

- request_get_bim_services_info XML parameters (on page 180)
- request_bim_services_info XML example (on page 180)
- response_get_bim_services_info XML parameters (on page 181)
- response_bim_services_info XML parameters example (on page 182)

request_get_bim_services_info XML parameters

The following table describes request_get_bim_services_info XML parameters:

Parameter	Description
user_token	Serial identification number supplied to the user during registration. String.
service_name	The name of the service.

For an example of a request, see request_bim_services_info XML example (on page 180).

request_bim_services_info XML example

The following example describes a successful request:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <ctmem:request_bim_services_info
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
            <ctmem:user_token>12345630</ctmem:user_token>
        </ctmem:request_bim_services_info>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

response_get_bim_services_info XML parameters

The following table describes response_get_bim_services_info XML parameters:

Parameter	Description
status	The status of the BIM service.
services	Information regarding the BIM service. See bim_services XML parameters (on page 182)

For an example of a response, see response_bim_services_info XML parameters example (on page 182).

NOTE: XML parameters for the get_bim_services_info parameter, as well as a sample fault response are described in Fault Response (on page 185).

bim_services XML parameters

The following table describes bim_services XML parameters:

Parameter	Description
order_date	The order date of the service.
service_name	The name of the service.
status	The status of the service.
due_time	The due time of the service.
min_end_time	The minimum time required for the service to complete.
description	The description of the service
num_jobs	The number of jobs contained in the service.
complete_percentage	The percentage of jobs completed.
jobs_completed	The number of jobs completed.
jobs_without_stats	The number of jobs without statistic information.
time_zone	Indicates the time zone according to which the job should be scheduled. When the value of this parameter is not zero, all dates reported use this time zone.
data_center	The data center where the service was ordered.
priority	The priority of the service.
min_slack_time	In case of a failure, this field indicates how long before the service is late.
status_color	Indicates the color of the status. Valid values are: ▪ Service in process or completed ▪ Job delay or Job finished early ▪ Service delay or will not complete ▪ Service completed late

response_bim_services_info XML parameters example

The following example describes a successful response:

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Body>
        <ctmem:response_bim_services_info
xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
            <ctmem:status>OK</ctmem:status>
            <ctmem:services>
                <ctmem:service>
                    <ctmem:odate>12:26:07 01/12/2014</ctmem:odate>
                    <ctmem:service_name>Bim Service</ctmem:service_name>
                    <ctmem:status>Service completed</ctmem:status>
                    <ctmem:due_time>12:29:07 01/12/2014</ctmem:due_time>
                    <ctmem:min_end_time>12:26:07
01/12/2014</ctmem:min_end_time>
                    <ctmem:description/>
                    <ctmem:num_jobs>3</ctmem:num_jobs>

<ctmem:complete_percentage>100</ctmem:complete_percentage>
                    <ctmem:jobs_completed>3</ctmem:jobs_completed>
                    <ctmem:jobs_without_stats>2</ctmem:jobs_without_stats>
                    <ctmem:time_zone> GMT +2</ctmem:time_zone>
                    <ctmem:data_center>palace620</ctmem:data_center>
                    <ctmem:priority>5</ctmem:priority>
                    <ctmem:min_slack_time>0</ctmem:min_slack_time>
                    <ctmem:status_color>Service in process or
completed</ctmem:status_color>
                </ctmem:service>
                <ctmem:service>
                    <ctmem:odate>12:22:00 01/12/2014</ctmem:odate>
                    <ctmem:service_name>Bim Service</ctmem:service_name>
                    <ctmem:status>Service is late</ctmem:status>
                    <ctmem:due_time>12:25:00 01/12/2014</ctmem:due_time>
                    <ctmem:min_end_time>13:52:08
01/12/2014</ctmem:min_end_time>
                    <ctmem:description>Job "Service4" should have started by
                        12:25:00 GMT+02:00 and will not start on time. The
reason
```

```
                        is: Job is held.</ctmem:description>
                    <ctmem:num_jobs>3</ctmem:num_jobs>

<ctmem:complete_percentage>33</ctmem:complete_percentage>
                    <ctmem:jobs_completed>1</ctmem:jobs_completed>
                    <ctmem:jobs_without_stats>2</ctmem:jobs_without_stats>
                    <ctmem:time_zone> GMT +2</ctmem:time_zone>
                    <ctmem:data_center>palace620</ctmem:data_center>
                    <ctmem:priority>5</ctmem:priority>
                    <ctmem:min_slack_time>0</ctmem:min_slack_time>
                    <ctmem:status_color>Service delay or will not
complete</ctmem:status_color>
                </ctmem:service>
            </ctmem:services>
        </ctmem:response_bim_services_info>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
<SOAP-ENV:Fault>
<faultcode> … </faultcode>
<faultstring>…. </faultstring>
<detail>
. . . FAULT_RESPONSE_OF_THE_REQUEST . . .
</detail>
</SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The fault parameters for all requests are described in Fault Response (on page 185).

Fault Response

The following table describes the parameters for fault XML parameters of all requests:

Parameter	Description
error_list	A sequence of error. For a list of the error_list sub parameters: ▪ **error attributes:** Includes the following: **Major**: The error Major Code. An integer describing the family of errors to which the error belongs. For more information, see Error codes and exceptions (on page 195) **Minor**: The error minor Code. An integer that is a unique ID for an error of this type. For more information, see Error codes and exceptions (on page 195) **Severity:** The priority level assigned to the error ▪ **error_message:** Text description of the error. For more information see Error codes and exceptions (on page 195).

For an example of a fault response, see Fault Example (on page 185).

Fault Example

```
<?xml version="1.0" encoding="iso-8859-1"?>
        <SOAP-ENV:Envelope
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
            <SOAP-ENV:Body>
                <SOAP-ENV:Fault>
                    <faultcode>SOAP-ENV:Server</faultcode>
                    <faultstring>Error response from EM
        Server.</faultstring>
                    <detail>
                        <ctmem:fault_def_create_jobs
        xmlns:ctmem="http://www.bmc.com/ctmem/schema900">
                            <ctmem:error_list
        ctmem:highest_severity='Error' >
                                <ctmem:error ctmem:major='412'
        ctmem:minor='1' ctmem:severity='Error' >
                                    <ctmem:error_message>
                                        Create jobs definitions failed,
        inavlid params.
```

```
                            </ctmem:error_message>
                        </ctmem:error>
                        <ctmem:error ctmem:major='412'
ctmem:minor='14' ctmem:severity='Error' >
                            <ctmem:error_message>
                                Create jobs definitions validation
error: 'Field: AuthorError: The
                                                    field must have a
value'.
                            </ctmem:error_message>
                        </ctmem:error>
                    </ctmem:error_list>
                </ctmem:fault_def_create_jobs>
            </detail>
        </SOAP-ENV:Fault>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

8

Diagnostics and troubleshooting

This topic helps you to analyze the Control-M/EM API product's performance and prevent conflicts before they arise. The following topics describe logging, configuration troubleshooting and application runtime and communication problems:

Control-M/EM API logging

Control-M/EM API logs actions, exceptions, and warnings. The logging mechanism uses the Log4J library from the Jakarta Java Solutions project. The manifest file located at **emapi-version\classes\log4j-1.2.8.jar** indicates which version of Log4J is in use.

The Control-M/EM API logging default behavior has the following characteristics:

- The Control-M/EM API logs only errors (priority level: ERROR).
- The maximum log file size is 50 KB.
- The Control-M/EM API saves the last two completed logs.
- Log files are rolling, not cyclical.

You can modify the logging procedures if you want to enable greater control over logging behavior. See Modifying logging behavior (on page 187) and the Log4J documentation.

Modifying logging behavior

This procedure describes how to modify logging behavior. Log default settings are coded into Control-M/EM API and these settings can be overridden by values supplied from an outside source.

You can supply additional parameters to extend logging capabilities. Any logging configuration information that you enter in the **emapi_log.cfg** file or in your project code must conform to Log4J specifications (as described in the Log4J documentation).

➢ To modify the logging procedure:

- Do one of the following:
  - Create a file called **emapi.log.cfg** in the working directory of your application or project.

    When this file is present, the Control-M/EM API automatically passes the information contained in the file to Log4J. For an example, see Sample log configuration file (on page 188).
  - Use your own code to pass information to Log4J. Control-M/EM API uses the `com.bmc.ctmem.emapi` logging category to log data.

If you are modifying logging behavior, you need to indicate the priority of messages (on page 188) recorded in the log in your configuration file.

Log message priorities

The following table describes the message priority levels supported by Log4J. The table orders the levels from the most critical to the least critical.

NOTE: Each setting includes messages of that level plus the more critical levels.

The following table describes Priority levels for log messages:

Level	Description
FATAL	displays fatal error messages
ERROR	displays messages for fatal and non-fatal errors
WARN	displays warning messages and all error messages
INFO	displays system information, warnings, and all errors
DEBUG	displays debugging information and all other priority messages

Sample log configuration file

The following example shows a sample log configuration file that contains common attributes.

emapi_log.cfg file example with default parameters

```
log4j.rootLogger=ERROR
log4j.category.com.bmc.ctmem.emapi=ERROR, EMAPI_Appender
# set appender for EM/API
log4j.appender.EMAPI_Appender=org.apache.log4j.RollingFileAppender
log4j.appender.EMAPI_Appender.file=emapi_log_file
log4j.appender.EMAPI_Appender.append=true
log4j.appender.EMAPI_Appender.maxFileSize=10kb
log4j.appender.EMAPI_Appender.maxBackupIndex=2
log4j.appender.EMAPI_Appender.layout=org.apache.log4j.SimpleLayout
```

The first line (`log4j.rootLogger=ERROR`) sets the priority of the root category to ERROR. If your project code uses the Log4J library, the root category determines the highest overall priority of log messages.

The second line defines an appender for a specified category. The category can have multiple appenders. The appender that is defined here is assigned a logging priority.

The appender is EMAPI_Appender.

The category is `com.bmc.ctmem.emapi.`

The logging priority is ERROR.

The remaining lines of the example define the properties of the appender, EMAPI_Appender. The valid properties depend on the type of appender that is being defined.

This example shows a RollingFileAppender that has the properties discussed in the following table.

RollingFileAppender example properties

Code	Description
file=emapi.log	indicates that the log output file is named **emapi.log**
append=true	indicates that new information is added to the end of the log file
maxFileSize=50kb	indicates that the maximum file of the log file is 50 KB
maxBackupIndex=2	indicates the number of backups made (that is, the number of old log files saved)
layout=org.apache.log4j.SimpleLayout	indicates the format for entries to the log file

Environment configuration troubleshooting

Modifications or omissions in the environment configuration can cause the Control-M/EM API to malfunction.

The following topics discuss the following problems and their solutions:

- CLASSPATH: missing libraries or directories (on page 189)
- Error in Java virtual machine parameters (on page 190)
- Incompatible object argument for a function call (on page 191)

CLASSPATH: missing libraries or directories

Null exception at some point during runtime may be because some Control-M/EM API directories, **.jar** libraries, or library locations could be missing from your application class path (CLASSPATH).

Corrective Action: Ensure that the following files and directories are in the class path:

- *fullPath***/emapi-900/classes/log4j-1.2.8.jar**
- *fullPath***/emapi-900/classes/emapi.jar**
- *fullPath***/emapi-900/classes/jbcl.jar**
- *fullPath***/emapi-900/classes/jacorb.jar**
- *fullPath***/emapi-900/classes/concurrent-1.3.2.jar**
- *fullPath***/emapi-900/classes/logkit-1.2.jar**
- *fullPath***/emapi-900/classes/commons-codec-1.3.jar**
- *fullPath***/emapi-900/classes/avalon-framework-4.1.5.jar**
- *fullPath***/emapi-900/classes/antlr-2.7.2.jar**
- *fullPath***/emapi-900/classes/xercesImpl.jar**
- *fullPath***/emapi-900/classes/xml-apis.jar**
- *fullPath***/emapi-900/etc**
- *fullPath***/emapi-900/etc/jacorb.properties**

Error in Java virtual machine parameters

Explanation: If your application fails during **init** initialization or when the **invoke** method is first used, there may be an error in the virtual machine's configuration parameters. Check the log file for a message similar to the following message:

```
ERROR - resolve naming service failed during initial reference:
org.omg.CORBA.COMM_FAILURE:   minor code: 1398079490  completed: No
at com.sun.corba.se.internal.iiop.IIOPConnection.writeLock(Unknown Source)
```

If you receive an **org.omg.CORBA.COMM_FAILURE** exception, and the exception was thrown from the **com.sun.corba.se.internal.iiop** package, you probably did not specify to the virtual machine that it must use the JacORB (CORBA) implementation (in place of the Sun default implementation).

Corrective Action: You must specify to the Java virtual machine that it must use the CORBA implementation.

Use one of the following methods to solve the problem:

- Run the java command from the project working directory, using the following parameters:

  ```
  java -Dorg.omg.CORBA.ORBClass=org.jacorb.orb.ORB
  -Dorg.omg.CORBA.ORBSingletonClass=org.jacorb.orb.ORBSingleton
  projectAppName
  ```

  projectAppName is the executable file of your project.

- Create a file containing the following text:

  ```
  org.omg.CORBA.ORBClass=org.jacorb.orb.ORB
  org.omg.CORBA.ORBSingletonClass=org.jacorb.orb.ORBSingleton
  ```

- Save the file with the name **orb.properties** in one of the following directories:
  - If you are using JDK, use *JDK_HOME***/jre/lib**
  - If you are using JRE, use *JRE_HOME***/lib**
- Call the **init** method by using the properties illustrated in the following code example (assuming that **args** is your application command-line arguments).

```
Properties props = new Properties();
props.setProperty("org.omg.CORBA.ORBClass",
"org.jacorb.orb.ORB");
props.setProperty("org.omg.CORBA.ORBSingletonClass",
"org.jacorb.orb.ORBSingleton");
EMXMLInvoker.init(args, props);
```

- Alternatively, call the **init** method without any parameters.

Incompatible object argument for a function call

Calls may fail when using EMBasicXMLInvoker, but succeed when using EMXMLInvoker if irrelevant libraries in the application class path (CLASSPATH) conflict with libraries that are required by the Control-M/EM API. For example, a newer or older version of the APACHE Xerces Parser might be found in the CLASSPATH.

Corrective Action: Use one of the following solutions:

- Use the smallest set of classes possible when running the API. Doing so prevents your project from finding irrelevant products and libraries in the CLASSPATH.
- Ensure that the you have listed the required libraries first in the CLASSPATH so that they are found first.

Application runtime and communication troubleshooting

Communication with Control-M/EM server components is essential to the operation of the Control-M/EM API. A disruption in communication can cause an exception to be thrown.

This section discusses the following problems and their solutions:

- An exception is thrown by the invoke method (on page 192)
- An error occurs when an XML file is submitted. (on page 192)
- Application cannot be started (on page 193)

An exception is thrown by the invoke method

An exception is thrown when you use the invoke method.

The following are possible causes:

- The Control-M/EM API cannot communicate with the CORBA Naming Service because either the CORBA services are down or the Naming Service host name and port settings in the **jacorb.properties** file are incorrect. You can find the corbaloc reference by searching for the following string: **ORBInitRef.NameService**. Ensure that the following requirements are met to solve the problem:
    - The CORBA Naming Service is running (started) on the CORBA Server host computer.
    - The Naming Service host name and port settings in the **jacorb.properties** file point to the computer that hosts the CORBA Server that you are using.
- Communication cannot be established with the Control-M/EM GUI Server because either the host names of the GUI Server are incorrect in the **ctmemapi.properties** file or host names of the GUI Server were specified with incorrect values in the code. Ensure that the following requirements are met to solve the problem:
    - The Control-M/EM GUI Server is running.
    - The host names and ports of the Control-M/EM GUI server components are registered correctly in the CORBA Naming Service.
    - If you are using more than one CORBA Server, you are currently using the CORBA Server that serves the Control-M/EM GUI Server to which you are sending Control-M/EM API requests

An error occurs when an XML file is submitted.

An error occurs when an XML file is submitted. This may be because the XML file contains characters that the XML standard considers invalid. The following characters are translated in the XML file as indicated in the following table:

Character	Meaning	Translate to
<	less than	<
>	greater than	>
&	ampersand	&
"	quotation marks	"
‘	apostrophe	'
ASCII 10	line feed
ASCII 13	carriage return

Application cannot be started

Applications based on Control-M/EM API cannot be started possibly because of Java limitations, applications that use the Control-M/EM API cannot be started when the path of the directory in which the Control-M/EM API is installed contains spaces or other special characters.

For solutions see Solution 1: Application cannot be started (on page 193) and Solution 2: Application cannot be started (on page 193).

Solution 1: Application cannot be started

This procedure describes how to solve the problem that your Application cannot be started (on page 193) because of Java limitations.

➢ To solve the problem:

1. Move the Control-M/EM API directory structure to a path that does not contain spaces or special characters.
2. Reconfigure the Control-M/EM API.
3. If needed, modify the **JAVA_HOME** and CLASSPATH environment variables.

   NOTE: *JAVA_HOME* refers to the directory where the Java 2 Runtime Environment (JRE) was installed. The Java 2 SDK (also called the JDK) contains the JRE, but at a different level in the file hierarchy.

EXAMPLE: If the Java 2 SDK or JRE was installed in **/ctm_em/user1**, *JAVA_HOME* would be as follows:

If you are using JRE, **/ctm_em/user1/jre1.6.x**

If you are using SDK, **/ctm_em/user1/jdk1.6.x/jre**

Solution 2: Application cannot be started

This solution provides an alternate method for use with the Control-M/EM API. If you implement this solution, use this method at all times.

➢ To solve the problem:

- **Do one of the following**
  - **UNIX:**
    - Define an alias to bypass the path that contains spaces. The alias should point to the Control-M/EM API **emapi-900** root directory. By using this alias when calling the Control-M/EM API, you can avoid referring to spaces and special characters that exist in the actual path.
    - After defining the alias, reconfigure the Control-M/EM API so that it uses the new path.

EXAMPLE: If the Control-M/EM API root directory is **ctm_em/path with space/emapi-900**, use the following commands to create the alias:

```
cd ctm_em
ln -s "path with space" pathwithoutspace
cd ctm_em/pathwithoutspace/emapi-900
```

EXAMPLE: Reconfigure the Control-M/EM API to reflect the new path.

- **Microsoft Windows:**
  - To avoid referring to spaces and special characters that exist in the actual path of the Control-M/EM API, call the Control-M/EM API by using the MS-DOS-compatible (short) 8.3-format version of its name. Ensure that you reconfigure the Control-M/EM API so that it uses the 8.3-format version of its path.

EXAMPLE: Open a new command prompt window.

Change the directory to the Control-M/EM API root directory, referring to the directory by using the 8.3-format version of its name.

If the Control-M/EM API root directory is **d:\Program Files\BMC Software\emapi-900**, use the following command:

cd progra~1\bmcsof~1\emapi-900

Reconfigure the Control-M/EM API to reflect the new path.

9

Error codes and exceptions

The following topics describe the errors and exceptions, which may occur when the Control-M/EM API is being used. Errors and exceptions can be divided into the following groups:

- Errors and exceptions that are caused by Control-M/EM API
- Errors and exceptions that are caused by faulty formatting in the XML request file
- Errors and exceptions that are caused by the presence of incorrect data in the XML request file
- Errors and exceptions that are caused by Control-M/EM or the Control-M installation

NOTE: Control-M/EM API errors are also listed in the **EMAPIMessages.txt** file located in the home directory of the Control-M/EM installation.

Each error is composed of the following information:

- **Major code**

  Integer that represents a family of related errors or exceptions.

- **Minor code**

  ID code unique to each error.

- **Severity** (on page 195)

  Indicates how critical the error is. Severity also determines the way that the Control-M/EM API handles the error.

- **Description**

  Text description of the error, returned as a string when an exception is thrown.

Severity

Every error that the Control-M/EM API displays has a severity level. This level indicates the priority of the error and indicates how much information is written to the log file.

When using the Control-M/EM API, the most common severity level is **ERROR**. Using the default logging configuration, the **DEBUG** error is never used. For information on how to modify the logging configuration, see Control-M/EM API logging (on page 187).

The severity levels that are supported by the Control-M/EM API are listed in the following table:

Level	Description
FATAL	Displays fatal error messages. **NOTE:** When a **FATAL** error is generated, it is recommended that you stop using the Control-M/EM API immediately, and that you investigate the cause of the error. If you continue to use Control-M/EM API despite receiving a **FATAL** error, stop and restart Control-M/EM GUI Server and Global Alerts Server before performing any more API actions.
ERROR	Displays messages for fatal and non-fatal errors
WARN	Displays warning messages and all error messages
INFO	Displays system information, warnings, and all errors
DEBUG	Displays debugging information and all other priority messages

Error code reference

All Control-M/EM API errors are arranged in logical groups. Each group is identified by a unique 3 - 5-digit Major Code. Each error in the group is identified by a unique 1 - 3-digit Minor Code.

The following table lists all of the Major Code groups and includes a reference to the pages where the errors in each group are described.

Individual errors are described in tables arranged by Major Code. The Minor Code, severity, and description are listed for each error in the tables.

Error and exception major codes

Code	Title	Error descriptions
000	NULL Exception	NULL exception errors (Major code 000) (on page 198)
100	Low-level API Exceptions	Low-level API exceptions (Major code 100) (on page 198)
200	Parser Exceptions	Parser exceptions (Major code 200) (on page 199)
300	Control-M Server Errors	**NOTE:** Category contains no individual errors.
300	Control-M Server Errors: Group1	Control-M/Server errors: Group 1 (Major code 300) (on page 199)
301	Control-M Server Errors: Group 2	Control-M/Server errors: Group 2 (Major code 301) (on page 202)
302	Control-M Server Errors: Group 3	Control-M/Server errors: Group 3 (Major code 302) (on page 203)
400	Control-M/EM API Request Exceptions	**NOTE:** Category contains no individual errors.
401	Generic Request Exceptions	Generic request exceptions (Major code 401) (on page 204)
403	Poll Request	Poll request errors (Major code 403) (on page 204)
404	Add/Delete Condition Request	Add or Delete Condition request errors (Major code 404) (on page 204)
405	Order/Force Request	Order or Force request errors (Major code 405) (on page 205)
406	Job Tracking Request	Job tracking request errors (Major code 406) (on page 206)
407	Authorization Request	Authorization request errors (Major code 407) (on page 206)
408	Alerts Request	Alerts request errors (Major code 408) (on page 207)
409	Create Active Job Request	Create active job request errors (Major code 409) (on page 207)
411	Upload folders	Upload folder request errors (Major code 411) (on page 208)
412	Create jobs or SMART Folder definitions	Create job/SMART Folder definitions request errors (Major code 412) (on page 208)

Code	Title	Error descriptions
413	Delete job definitions	Delete job definitions request errors (Major code 413) (on page 209)
440	Retrieve active jobs	Retrieve active jobs request errors (Major code 440) (on page 210)
450	Active job actions	Job actions request errors (Major code 450) (on page 210)
500	Control-M/EM API Java Client	Control-M/EM API Java client errors (Major code 500) (on page 211)
600	Gateway Messages	Gateway messages (Major code 600) (on page 212)

NULL exception errors (Major code 000)

Control-M/EM API has generated an undefined error.

The following table describes the NULL exceptions:

Minor code	Severity	Description
0	ERROR	Control-M/EM API has generated an undefined error

Low-level API exceptions (Major code 100)

Errors generated due to a Control-M/EM API initialization or request initialization failure.

The following table describes the Low level API exceptions:

Minor code	Severity	Description
1	ERROR	Error on preliminary parsing of the XML
2	FATAL	Catastrophic exception on server
3	ERROR	Error: *value* Rule Based Calendar missing
4	ERROR	Invalid Request. Request name: *value*
5	FATAL	Could not obtain response from repository

Minor code	Severity	Description
6	ERROR	Undefined exception on server
7	ERROR	Deprecated request

Parser exceptions (Major code 200)

Error occurred when parsing an XML file.

The following table describes the Parser exceptions.

Minor code	Severity	Description
1	FATAL	Parser Initialization Failure
2	ERROR	Error while parsing XML: *value*
3	FATAL	Internal Parser Error: *value*
4	FATAL	Internal Parser Error: Undefined Exception

Control-M/Server errors: Group 1 (Major code 300)

Errors generated by the Control-M installation.

The following table describes the Control-M/Server errors Group 1:

Minor code	Severity	Description
0	ERROR	Internal Server Error on Control/M
3	ERROR	Control/M Error: *value*
5	ERROR	Storage allocation failed for CTM/EM Gateway
7	ERROR	Cannot open conditions file
9	ERROR	Loading of Control-M Parameters failed
18	ERROR	Internal error on GETMEM

Minor code	Severity	Description
19	ERROR	DSN not in catalog
20	ERROR	DSN - dynamic allocation failed
21	ERROR	Internal error. Invalid request to CTMMEM
22	ERROR	Maximum number of members or lines in member exceeded
23	ERROR	Invalid return code from CTMMEM
24	ERROR	Error while processing directory of library
25	ERROR	Library operation failed
40	ERROR	Cannot open print file
42	ERROR	Open of IOA log file failed
45	ERROR	Internal Error on IOACND
46	ERROR	Invalid date format
47	ERROR	Cannot open synchronization file
48	ERROR	Internal Server Error on Control-M
49	ERROR	Internal Server Error on Control-M
51	ERROR	Cannot add the condition because it already exists
52	ERROR	Cannot add the condition because the condition file is full
53	ERROR	Cannot add the control resource because it already exists
54	ERROR	Cannot add the control resource because the resource file is full
55	ERROR	Cannot delete the condition because it does not exist
56	ERROR	Cannot delete the quantitative resource because it is in use
57	ERROR	Cannot add the quantitative resource because the resource file is full
58	ERROR	Cannot add the quantitative resource because it already exists
59	ERROR	Cannot delete the quantitative resource because it does not exist

Minor code	Severity	Description
60	ERROR	Change command can only be issued against quantitative resources
61	ERROR	Invalid value in sign field
62	ERROR	The value in a change command must be between 1-9999
63	ERROR	Cannot load module (internal error)
64	ERROR	The requested folder does not exist in the library
70	ERROR	The requested job does not exist in the given folder
87	ERROR	Invalid date
90	ERROR	Folder or Member does not exist
91	ERROR	Folder or Member already exists
93	ERROR	Invalid Folder ID
94	ERROR	The 'NEWG' order option is not supported in this version
95	ERROR	The folder entity specified was not found in Active Jobs database
96	ERROR	Failed to extract data from CTM/EM message
97	ERROR	Failed to delete records from folder
98	ERROR	Insert into folder failed
99	ERROR	Update folder failed
100	ERROR	Failed to commit transaction
101	ERROR	The filed Sub Application is mandatory for a group entity
102	ERROR	SUB APPLICATION and JOBNAME should be the same for group entity
103	ERROR	SUB APPLICATION and SCHEDTAB should be the same for group entity
104	ERROR	SUB APPLICATION and SCHEDTAB should be specified
105	ERROR	The Sub Application specified does not exist in the database
106	ERROR	FILE_PATH, CMDLINE, FILE_NAME AND OVERRIDE_PATH should not be specified

Minor code	Severity	Description
107	ERROR	Failed to allocate ISN
108	ERROR	No hosts found in host group
109	ERROR	Failed to get next host in host group
110	ERROR	Failed to setup application type for HOSTGRP, FILE_NAME and Override Path should not be specified
111	ERROR	Invalid Order_Date

Control-M/Server errors: Group 2 (Major code 301)

Errors generated by the Control-M installation.

The following table describes the Control-M Server errors Group 2:

Minor code	Severity	Description
501	INFO	The job has been successfully ordered
502	ERROR	Unable to open the specified scheduling library
506	ERROR	Scheduling failed for member
510	ERROR	Scheduling member contains invalid data
512	WARN	Library should be compressed
514	ERROR	Job was not ordered. Reason: Insufficient storage
515	ERROR	Job contains too many cards
516	ERROR	Table error: First card is not a D statement
517	ERROR	The specified library is empty
524	ERROR	Ordering process has entered with errors
525	ERROR	Ordering process ended successfully
526	ERROR	Invalid data format

Minor code	Severity	Description
528	INFO	Job has been successfully ordered
531	ERROR	Ordering process was canceled by an user exit
532	ERROR	Cannot open Control-M/EM Active Jobs database
534	ERROR	Cannot open Active Jobs database is corrupted, I/O error, or file is not really Active Jobs
535	ERROR	Cannot order a job while Active Jobs database is being formatted
536	ERROR	Severe error in scheduling definition
537	ERROR	The job order contains more information than what Control-M/EM can handle
548	ERROR	The calendar specified in the job is either corrupted or invalid
549	WARN	Control-R is not installed. IFRERUN statement ignored
550	WARN	Control-R is not installed. SET statement ignored
166	ERROR	The job contains a condition with a PREV/NEXT date that cannot be interpreted by Control-M/EM
169	INFO	Control-M/EM has finished handling the request. No jobs were scheduled

Control-M/Server errors: Group 3 (Major code 302)

Errors generated by the Control-M installation.

The following table describes the Control-M Server errors Group 3:

Minor code	Severity	Description
129	ERROR	User exit not loaded
540	ERROR	Security exit not loaded and security checking is bypassed
863	WARN	Active Jobs database is nearly full

Generic request exceptions (Major code 401)

Errors generated when Control-M/EM cannot communicate with the Control-M installation.

The following table describes the Generic request exceptions:

Minor code	Severity	Description
1	ERROR	Could not connect to Control-M
2	ERROR	Invalid Control-M
3	ERROR	Invalid response from Control-M
4	ERROR	Internal Error: *value*

Poll request errors (Major code 403)

Errors generated when a Polling request fails.

The following table describes the Poll request errors:

Minor code	Severity	Description
1	ERROR	Error: Invalid token supplied

Add or Delete Condition request errors (Major code 404)

Errors generated when an Add Condition or Delete Condition request fails.

The following table describes the Add or Delete Condition request

Minor code	Severity	Description
1	ERROR	Add or Delete condition failed (code %d)
2	ERROR	Add or Delete condition failed
3	ERROR	Add or Delete condition aborted
4	ERROR	Add or Delete condition timed out
5	ERROR	Add condition failed

Minor code	Severity	Description
6	ERROR	Add condition failed (code %d)
7	ERROR	Add condition failed, invalid option
8	ERROR	Delete condition failed
9	ERROR	Delete condition failed (code %d)
10	ERROR	Delete condition failed, invalid option
11	ERROR	Condition's name is not valid
12	ERROR	Condition's order date is not valid
13	ERROR	Cannot Add or Delete condition, already in wanted state

Order or Force request errors (Major code 405)

Errors generated when an Order request or Force request fails.

The following table describes the Order/Force request errors:

Minor code	Severity	Description
1	ERROR	Order request didn't pass validity checks. Error: *value*
2	ERROR	Group RBA not found for Group ID *value*
3	ERROR	Order request failed in the server
4	ERROR	No jobs were ordered

Job tracking request errors (Major code 406)

Errors generated when a Job tracking request fails.

The following table describes the Job tracking request errors:

Minor code	Severity	Description
1	ERROR	Job was not found in Active Jobs database
2	ERROR	Required Job information does not exist in the database
3	ERROR	Tracked Job failed in validity checks

Authorization request errors (Major code 407)

Errors generated when a registration or unregistration request fails.

The following table describes the Authorization request errors:

Minor code	Severity	Description
1	ERROR	Invalid user token
2	ERROR	Invalid user name
3	ERROR	Register failed
4	ERROR	Unregister failed
5	ERROR	User not authorized
6	ERROR	Account is locked
7	ERROR	Password has expired

Alerts request errors (Major code 408)

Errors generated when a Change Alert Status request fails.

The following table describes the Alerts request errors:

Minor code	Severity	Description
1	ERROR	Alert id is not valid
2	ERROR	Invalid alert operation

Create active job request errors (Major code 409)

Errors generated when a Job Creation request fails.

The following table describes the Create active job request errors:

Minor code	Severity	Description
1	ERROR	Create active job failed (code: %d)
2	ERROR	Create active job failed
3	ERROR	Create active job failed, object is in use
4	ERROR	Create active job failed, object not found
5	ERROR	Create active job failed, invalid option
6	ERROR	Create active job validation error: *value*
7	ERROR	SMART Folder is not valid for this data-center version
8	ERROR	Failed to initialize job descriptor
9	ERROR	Create active job, invalid order date

Upload folder request errors (Major code 411)

Errors generated when an Upload Folder request fails.

The following table describes the Upload folder request errors:

Minor code	Severity	Description
1	ERROR	Upload folder failed, invalid parameters
2	ERROR	Upload folder failed
3	ERROR	Cannot get folder from database
4	ERROR	Upload folder aborted
5	ERROR	Upload folder timed-out
6	ERROR	Upload folder failed with status

Create job/SMART Folder definitions request errors (Major code 412)

Errors generated when a Create job/SMART Folder definitions request fails.

The following table describes the Create job/SMART Folder definitions request errors:

Minor code	Severity	Description
1	ERROR	Create job definitions failed, invalid parameters
2	ERROR	Create folder failed
3	ERROR	Create job definitions failed
4	ERROR	Failed to add jobs to folder
5	ERROR	Create jobs definitions failed, unknown reason
6	ERROR	<TBD: Message text>
7	ERROR	Folder already exists
8	ERROR	SMART Folder is not valid for this data-center version

Minor code	Severity	Description
9	ERROR	Failed to initialize job descriptor
10	ERROR	Create definitions, API is not supported for CTM version '%s'
11	ERROR	SMART Folder already contains a sub-folder
12	ERROR	Illegal author in new job
13	ERROR	Sub application name of a job or a SMART Folder differs from a SMART Folder or folder
14	ERROR	Create jobs definitions validation error: '%s'

Delete job definitions request errors (Major code 413)

Errors generated when a Delete job definitions request fails.

The following table describes the Delete job definitions request errors:

Minor code	Severity	Description
1	ERROR	Folder does not exist
2	ERROR	Many folders are found
3	ERROR	Failed to delete jobs from folder
4	ERROR	No jobs were deleted according to the specified criterion
5	ERROR	Failed to update number of jobs in a SMART Folder
6	ERROR	Delete jobs definitions validation error: [<*filter name*>] is not valid filter field
7	ERROR	Delete jobs definitions, invalid parameters

Retrieve active jobs request errors (Major code 440)

Errors generated when a Retrieve active jobs request fails.

The following table describes the Retrieve active jobs request errors:

Minor code	Severity	Description
1	ERROR	Retrieve active jobs failed
2	ERROR	Request failed - internal error
3	ERROR	No jobs were found according to the specified criterion
4	ERROR	Partial result
5	ERROR	The maximum of returned nodes, which is specified in request, exceeds Control-M/EM server limit, which is specified by EMAPIActiveJobsLoadLimit system parameter
6	ERROR	Retrieve active jobs validation error: '%s' is not valid filter field

Job actions request errors (Major code 450)

Errors generated when a Job actions request fails.

The following table describes the Job actions request errors:

Minor code	Severity	Description
1	ERROR	Unknown error occurred
2	ERROR	Action request failed (code %d)
3	ERROR	Action request failed
4	ERROR	Action request aborted
5	ERROR	Action request timed-out
304	ERROR	Cannot hold job - not held
305	ERROR	Active Jobs database is locked, try again later
306	ERROR	Job does not exist

Minor code	Severity	Description
309	ERROR	Cannot hold job - already held
310	ERROR	Job is not waiting for confirmation
323	ERROR	Security protection violation
328	ERROR	Job is not in execution state
361	ERROR	The selected job/folder cannot be deleted due to its current state
506	ERROR	Operation not supported by agent
305	ERROR	Job does not exist
305	ERROR	Job actions, invalid order date

Control-M/EM API Java client errors (Major code 500)

Errors generated by the Java API.

The following table describes the Control-M/EM API Java client errors:

Minor code	Severity	Description
1	ERROR	Fatal error
2	ERROR	Null XML document - nothing to do
3	ERROR	Failed to establish connection with server - no server registered under this hostname
4	ERROR	Failed to resolve server name - please check your hostname
5	ERROR	Failed to establish connection with server - not a valid component type, check your hostname
6	ERROR	Failed resolving XMLInvoker interface
7	ERROR	Failed to establish connection with server - check your connection with the CORBA Naming Service
8	ERROR	Failed to establish connection with - check your CORBA configuration

Minor code	Severity	Description
9	ERROR	Properties file not found
10	ERROR	Failed to read properties file
101	ERROR	Invoke timeout - no response after
102	ERROR	Invoke request exited because a InterruptedException occurred
111	ERROR	Response format is not valid
112	ERROR	Response format is not valid, cannot find rule based calendar
131	ERROR	XML format is not valid
132	ERROR	Request format is not valid
133	ERROR	Cannot find user token
140	ERROR	Error parsing the XML
141	ERROR	Undefined Exception on Parser

Gateway messages (Major code 600)

Errors generated by the gateway.

The following table describes the Gateway messages:

Minor code	Severity	Description
1	ERROR	User Request timed out
2	ERROR	No network currently loaded
3	ERROR	No nodes match the Show Net parameters
4	ERROR	Net too large to be fully viewed. Some nodes are missing
5	ERROR	Database login failed three times
6	ERROR	At least one state must be chosen
7	ERROR	At least one status must be chosen

Minor code	Severity	Description
8	ERROR	At least one task type must be chosen
9	ERROR	No events found for current net
10	ERROR	Field *value* Wrong Format: *value*
11	ERROR	Cannot load Active Network: No data center
12	ERROR	Field *value* is required
13	ERROR	Net Load aborted, probably not enough memory
14	ERROR	No nodes match the Load Net Parameters
15	ERROR	Error while reading job-record from database
16	ERROR	Nothing changed since last save
17	ERROR	Nothing changed
18	ERROR	Cannot save file *value*
19	ERROR	Confirm *value* for *value*
20	ERROR	Do you really want to quit *value*?
21	ERROR	In a critical job, parameter 'Priority' must begin with
22	ERROR	Field *value* contains an invalid value
23	ERROR	Field *value* must be between %d and %d
24	ERROR	Field Rerun Mem Cannot be used if field Task Type is Cyclic
25	ERROR	Field *value* has an invalid value starting at position: *value*
26	ERROR	Field *value* may not contain embedded spaces
27	ERROR	Cannot clear Demo Net *value*
28	ERROR	Cannot create Demo Net *value*
29	ERROR	Cannot copy Demo Net *value*
30	ERROR	Couldn't open user view

Minor code	Severity	Description
31	ERROR	Do you really want to erase the entire *value* Net *value*
32	ERROR	Do you really want to clear the entire *value* Net *value*
33	ERROR	Cannot append, Owner of *value* is *value*
34	ERROR	Cannot open *value*
35	ERROR	Missing file *value*
36	ERROR	Unable un-mount diskette in *value*
37	ERROR	Unable to mount diskette in *value*
38	ERROR	Incorrect path *value*
39	ERROR	Files downloaded from incompatible version *value* (should be *value*)
40	ERROR	Cannot open file *value*
41	ERROR	Missing *value* line
42	ERROR	Unknown line: *value*
43	ERROR	Cannot write to file *value*
44	ERROR	File *value* exists. Ok to overwrite?
45	ERROR	Net *value* does not exist
46	ERROR	Copy from file to file not allowed
47	ERROR	Database job update failed
48	ERROR	Database select failed for resource events
49	ERROR	Database select failed for override table, simulation stopped
50	ERROR	No current net
51	ERROR	No nodes were found
52	ERROR	No more nodes were found Go Back?
53	ERROR	No more nodes were found

Minor code	Severity	Description
54	ERROR	You are not authorized to perform this action
55	ERROR	If field *value* (Days/Week days Calendar) begins with ALL then the rest of the parameter must be blank
56	ERROR	Field Days Calender is required if field *value* contains the value: *value*
57	ERROR	Field Week Calender is required if field *value* contains the value: *value*
58	ERROR	Field Days has an invalid value starting at position: *value*; There should be a comma there
59	ERROR	Discard changes made in form?
60	ERROR	Change of calendar type destroys existing data. Proceed?
61	ERROR	Change of data center platform destroys existing data. Proceed?
62	ERROR	Data center unknown, please re-enter value
63	ERROR	Reading calendar failed - database problems
64	ERROR	Calendar is already in use by user *value*, try later
65	ERROR	Calendar write failed - database problems
66	ERROR	Unknown platform. Verification of data defaults to MVS. Do you want to continue?
67	ERROR	Calendar was modified. Proceed with download?
68	ERROR	Calendar *value* was downloaded successfully from data center *value*
69	ERROR	Calendar *value* was deleted successfully from data center *value*
70	ERROR	Calendar is in use by user *value*, and cannot be deleted"
71	ERROR	Confirm Calendar Delete"
72	ERROR	Confirm Calendar Upload"
73	ERROR	Forced Calendar Upload overrides calendar content in data center Proceed?
74	ERROR	You are not authorized to access the Calendar List window
75	ERROR	Filed *value* must be specified if field *value* is File, New Dest or Change Class

Minor code	Severity	Description
76	ERROR	Attempt to enter conflicting conditions
77	ERROR	Field *value* may only be used with the Change Class option
78	ERROR	Field *value* is too long
79	ERROR	Field Sign may not be + if the ODATE field value is **** or $$$$
80	ERROR	Unknown or disabled Platform! Form is opened as MVS!
81	ERROR	In a critical job, field Priority must begin with *
82	ERROR	Field Command Line may be specified only if Task Type is Command
83	ERROR	Field should be %d char(s) long for this Sysout option
84	ERROR	Field *value* cannot be used if Task Type is *value*
85	ERROR	Field *value* cannot specify GENREAL or USER=
86	ERROR	Fields Parm and From Class must be specified if field Option is Change Class
87	ERROR	Field Parm must be specified if field Option is New Dest or File
88	ERROR	Field Confirmation Cal cannot be used with fields PDS Name or Minimum
89	ERROR	Field Months cannot be used with fields PDS Name, Minimum or Dates
90	ERROR	Field Dates cannot be used with parameters PDS Name, Minimum, Months, Days, or Days Calendar
91	ERROR	Field Dates cannot be used with fields Week Days, Weeks Days Calender Confirmation Cal
92	ERROR	If field PDS Name is specified then Minimum must be specified and vice versa
93	ERROR	Field Priority must begin with * or an alphanumeric character
94	ERROR	Field Priority must begin with an alphanumeric character
95	ERROR	Field *value* is required if Task Type is: *value*
96	ERROR	A job with the same name already exists in this folder
97	ERROR	If field Retro is specified then PDS Name and Minimum cannot be used

Minor code	Severity	Description
98	ERROR	The fields Days and Week Days Calendar cannot both be specified
99	ERROR	The field *value* must contain a numeric value or 0
100	ERROR	Fields Rerun Mem and Max Rerun cannot be used if field Task Type is Cyclic
101	ERROR	Field Dates must be specified in format nnnn or nnnn,nnnn,...
102	ERROR	Folder is already in use by user *value*, try later
103	ERROR	Folder write failed - database problems
104	ERROR	Folder should be uploaded before order/force
105	ERROR	Field *value* is required
106	ERROR	If field *value* is *???????, then field *value* must be blank
107	ERROR	Folder was modified. Proceed with download?
108	ERROR	Folder was modified. Proceed with order?
109	ERROR	Folder was modified. Proceed with force?
110	ERROR	Definition folder *value* was downloaded successfully from data center *value*
111	ERROR	Definition folder *value* was deleted successfully from data center *value*
112	ERROR	Confirm Folder Delete
113	ERROR	Confirm Folder Upload
114	ERROR	Forced Folder Upload overrides folder content in data center Proceed?
115	ERROR	You are not authorized to access the Folders window
116	ERROR	Folder is in use by user *value*, and cannot be deleted
117	ERROR	Field *value* must be blank if field *value* is OK or NOTOK or RERUN
118	ERROR	Communication with WS-Gateway stopped, all requests were canceled
119	ERROR	System Error in *value*: Command *value* failed; *value*
120	ERROR	No communication with WS-Gateway. Request cancelled

Minor code	Severity	Description
121	ERROR	*value* not possible, no communication with Gateway
122	ERROR	*value* not possible, Unknown data center *value*
123	ERROR	*value* not possible, data center *value* not available
124	ERROR	Communication with data center *value* down, all requests canceled
125	ERROR	Data center *value* is suspended (Active jobs is being formatted)
126	ERROR	Data center *value* is suspended (Download in Netgroup)
127	ERROR	Data center *value* is suspended (Active jobs is being formatted), all requests canceled
128	ERROR	Invalid event %c received from data center *value*. To ensure database integrity, WS-GTW stop/restart is recommended
129	ERROR	The host name be specified only for the TCP protocol
130	ERROR	The host name may contain only Alphanumeric characters and periods
131	ERROR	You must specify a communication protocol
132	ERROR	Specified communication protocol not supported on platform
133	ERROR	Control-R not supported on platform
134	ERROR	The Net Group field must contain two alpha numeric characters
135	ERROR	Control-M must be 300 or 400 for MVS
136	ERROR	Incorrect Control-M version specified for this platform
137	ERROR	Folder read failed - database problems
138	ERROR	Please verify that all users have quit Control-M/EM and that the WS-Gateway is down.
139	ERROR	Do you really want to quit Control-M/Enterprise Manager Administration?
140	ERROR	No Net
141	ERROR	Number of nodes in a row exceeds maximum of %d
142	ERROR	Cannot generate report due to insufficient memory

Minor code	Severity	Description
143	ERROR	Cannot print links: *value*
144	ERROR	Cannot print index: *value*
145	ERROR	Cannot do poster for routed net
146	ERROR	Job Report fields saved. Please close and open the job report window
147	ERROR	New field is empty
148	ERROR	Confirm delete
149	ERROR	No Loaded Networks Available
150	ERROR	Control Resources with *value value* cannot be *value*
151	ERROR	Quantitative Resources with *value value* cannot be *value*
152	ERROR	Field Author must be entered
153	ERROR	Order ID *value* not found
154	ERROR	Pattern *value* not found in text
155	ERROR	Pattern *value* appears only once in the text
156	ERROR	Problems searching for pattern *value*
157	ERROR	Alarm handling command ignored. User *value* is handling this alarm now
158	ERROR	Alarm handling command ignored. Internal error
159	ERROR	Download of calendar *value* canceled in gateway - calendar in use
160	ERROR	Download of calendar *value* canceled in gateway - Another download is currently in progress - Try later
161	ERROR	Upload of calendar *value* canceled in gateway
162	ERROR	Upload of calendar *value* canceled in gateway - calendar in use
163	ERROR	Upload of definitions folder *value* canceled in gateway
164	ERROR	Upload of definitions folder *value* canceled in gateway - folder in use
165	ERROR	Download of definition folder *value* canceled in gateway - folder in use

Minor code	Severity	Description
166	ERROR	Download of definition folder *value* canceled in gateway - Another download is currently in progress - Try later
167	ERROR	Definitions folder deleted on data center but delete failed on workstation (rc 1)
168	ERROR	Definitions folder deleted on data center but delete failed on workstation (rc 2)
169	ERROR	Calendar deleted on data center but delete failed on workstation (rc 2)
170	ERROR	Delete of definitions folder *value* canceled in gateway - folder in use
171	ERROR	Delete of calendar *value* canceled in gateway - calendar in use
172	ERROR	User request canceled in gateway, no answer from data center (time-out)
173	ERROR	Download of *value* canceled, no answer from data center (time-out)
174	ERROR	Download of *value* canceled in gateway
173	ERROR	Download canceled, no answer from data center (time-out)
174	ERROR	Download canceled in gateway
175	ERROR	Upload canceled in gateway
176	ERROR	Upload canceled (file transfer error (rcp) in gateway), consult system administrator
177	ERROR	Information request canceled in gateway, system error (fopen failed), consult system administrator
178	ERROR	Information request canceled (time-out) in gateway, no answer from data center
179	ERROR	Upload of *value* canceled (time-out), no answer from data center
180	ERROR	Upload of *value* canceled in gateway
179	ERROR	Upload canceled (time-out), no answer from data center
180	ERROR	Upload canceled in gateway
181	ERROR	No fields defined
182	ERROR	Field *value* not in folder
182	ERROR	Field not in folder

Minor code	Severity	Description
183	ERROR	Internal error on field *value*
183	ERROR	Internal error on field
184	ERROR	Cannot print %d characters; Maximum of %d allowed
185	ERROR	Cannot print more than %d fields
186	ERROR	You are not authorized to load an active network
187	ERROR	*value* was not found in data center. Delete in workstation?
188	ERROR	You are not authorized to access the Global Conditions window
189	ERROR	Definition folder *value* was deleted only on workstation
190	ERROR	Calendar *value* was deleted only on workstation
191	ERROR	Internal error. Action failed
192	ERROR	Insert failed. Record with same key already exists
193	ERROR	Record deleted in the meantime
194	ERROR	Error message *value* Received from data center *value*
194	ERROR	Error message received from data center
195	ERROR	Field *value* is required if field *value* is defined
195	ERROR	Field required
196	ERROR	Field *value* should be at most %d characters
197	ERROR	Field *value* may not contain the characters *value*
198	ERROR	Field *value* must be between *value* and *value*
199	ERROR	Field *value* should be empty
200	ERROR	Field *value* must be blank if field When is OK or NOTOK or RERUN
201	ERROR	Field *value* must be *value*
202	ERROR	No response to heartbeat check from data center: *value*

Minor code	Severity	Description
202	ERROR	No response to heartbeat check from data center
203	ERROR	Unsupported version *value* of data center *value*. Check Control-M installation
203	ERROR	Unsupported version of data center, check Control-M installation
204	ERROR	Not the smallest ISN for ordered/forced job