IBM Tivoli Storage Manager

Client Management Services

REST API Guide

Document version [1.1]

Martine Wedlake
Senior Technical Staff Member, IBM

About the Tivoli field guides

Purpose

The Field Guide program has the following purposes:

- To empower customers & business partners to succeed with Tivoli software by documenting and sharing product information that provides accurate and timely information on Tivoli products and the business issues that impact an enterprise systems management project
- To leverage the internal knowledge within Tivoli Customer Support and Services and the external knowledge of Tivoli customers and Business Partners

Availability

Field guides are available free of charge to registered customers and internal IBM employees at the following website:

http://www.ibm.com/software/sysmgmt/products/support/Field_Guides.html

Authors can submit proposals and access papers by e-mail:

mailto:Tivoli_eSupport_Feedback@us.ibm.com

Table of Contents

REST API GUIDE I

Purpose ..... *iv*
Availability ..... *iv*

GETTING STARTED — A QUICK TUTORIAL ON USING CLIENT MANAGEMENT SERVICE 2

INSTALLING AND CONFIGURING CLIENT MANAGEMENT SERVICE 2
CREATING CLIENT MANAGEMENT SERVICE REST API APPLICATIONS WITH CODE EXAMPLES 3
Querying log records (GET example) *3*
Initiating backup (POST example) *5*

THE CLIENT MANAGEMENT SERVICE REST API FRAMEWORK 6

SECURITY CONSIDERATIONS 6
RESPONSE AND REQUEST MESSAGE BODIES 6
PAGING REQUESTS 6
FILTERING LOG RECORDS 7
LONG-RUNNING TASKS 7
ERROR RESPONSE 8
CLIENT MANAGEMENT SERVICE METADATA FOR CLIENTS 9
EXTENSIONS 9
LIMITATIONS AND KNOWN ISSUES 10

REST INTERFACE REFERENCE 11

ABOUT 11
BACKUPSYSTEMS 12
DAEMON 14
EXTENSIONS 16
TASKS 17

localObjects 19
savedObjects 21
configs: 23
logRecords: 24
Use the CmsConfig command to enable capabilities in the client management services for REST API. 26
CmsConfig enable command 26
CmsConfig disable command 27
CmsConfig setscriptdir command 28

NOTICES 30

Trademarks 31

Tivoli Storage Manager client management services

The client management services is an optional component of the IBM Tivoli Storage Manager Operations Center. The client management service is a Representational State Transfer (REST) service that is installed directly on the client computer. The Tivoli Storage Manager Operations Center uses the REST service to fetch client error and schedule logs to be displayed within the graphical user interface. This document is intended for customers, managed service providers, and vendors who want to create applications using the client management service REST application-programming interface (API). For example, a customer can use the REST API as an alternative to scripting for automation.

The client management service has the following capabilities available through a REST API:

- Read log files such as dsmerror.log and dsmsched.log.
- Perform backup and restore actions.
- Read and Update the client configuration files such as `dsm.opt` and `dsm.sys`.
- Read and update the client daemons such as client acceptor daemon and scheduler.
- Extend the REST resource namespace with extensibility scripts.

Tip: Beginning with Version 7.1.3, IBM Tivoli Storage Manager is now IBM Spectrum Protect. Some applications such as the software fulfillment systems and IBM License Metric Tool use the new product name. However, the software and its product documentation continue to use the Tivoli Storage Manager product name. To learn more about the rebranding transition, see http://www.ibm.com/support/docview.wss?uid=swg21963634.

1

Getting Started — A quick tutorial on using client management service

For security reasons, the client management service is limited to fetching error and schedule logs. You must configure client management service to enable these special features.

To get you started, you should understand the basic principles by using the tutorial program written in Python. The Python program shows you how to access the client management service functions. You are not required to use Python; any language that is capable of accessing REST services will work. Python was chosen because it is simple and fairly popular.

Installing and configuring client management service

To be able to use client management service, you must install client management service on the same computer as your backup-archive client. The client management service is delivered as a component to the Tivoli Storage Manager Operations Center. From the Operations Center, you can find instructions and download packages for client management service.

After you install client management service, you must configure client management service to use the enhanced capabilities. To change a capability, use the **CmsConfig** command (found in the *installDir*/cms/bin directory) to enable or disable the capability for a specified node.

The following table lists the full set of client management service capabilities.

client management service capability name	Description
LOG_QUERY	Query for error and schedule logs for the backup system.
EXTENSION_RUN	Request a list of available extensions and run them as a REST API (GET, PUT, POST, DELETE).
CONFIG_QUERY	Read configuration files.
CONFIG_UPDATE	Update configuration files.
DAEMON_QUERY	Read the daemon status (CAD or scheduler daemon).
DAEMON_UPDATE	Update daemons (start, stop, or restart the CAD or scheduler).
FILE_BACKUP	Read file information from local storage and request backups.
FILE_RESTORE	Read file information for existing backups and restore files.

For example:

```
CmsConfig.bat enable testnode FILE_BACKUP
```

The full syntax of the enable (and disable) commands can be found in CmsConfig .

Creating client management service REST API applications with code examples

Python is a good example of a fairly simple-to-use scripting language with REST modules available. The following examples use Python to demonstrate the GET and POST requests. To keep things as simple as possible, the examples use the built-in `http.client` package.

Requirements:

- Client management service REST API requires TLS 1.2 for enhanced security.
- You must use Python version 3.4 or later to run these scripts.

Querying log records (GET example)

In the following sample code, you must edit the following parameters for your specific solution:

- Replace ***`user`*** and ***`password`*** in the credentials with your admin user name and password.
- Replace ***`nodename`*** in the request URL with your node name.

```
import ssl
import http.client
import json
import base64

# Create an HTTPS connection to the  client management service. By default,
client management service
# installs a self-signed certificate so we have to pass in the special
# context parameter to avoid invalid certificate errors.

c = http.client.HTTPSConnection("localhost", 9028,
    context=ssl._create_unverified_context())

# Create headers to pass in the basic authentication information.

credentials=base64.b64encode(b"user:password").decode("ascii")
headers= {"Authorization" : "Basic " + credentials,
     "Accept"    : "application/json"}

# Issue the GET request against the log record resource.
# The node name is embedded in the URL.

c.request("GET", "/tsmcms/backupSystems/nodename/logRecords”,
    headers=headers)

# Fetch the response object. The response object can return all
# of the following information: response body, headers, status, reason, etc.

r = c.getresponse()

# Read the response body from the response object and decode it by using
# payloads in JSON, which uses UTF-8 encoding.

j = json.loads(r.read().decode('utf-8'))

# Dump the result to STDOUT
print (j);
```

Initiating backup (POST example)

```
import ssl
import http.client
import json
import base64

# Create an HTTP connection object
# Add a header for the content type.

c = http.client.HTTPSConnection("localhost", 9028,
     context=ssl._create_unverified_context())

credentials=base64.b64encode(b"user:password").decode("ascii")
headers= {"Authorization" : "Basic " + credentials,
     "Content-type" : "application/json",
     "Accept": "application/json"}

# Construct the request payload as a Python dictionary literal
# and then use json.dumps() to serialise it for use in the API.

backup = {
       "backupObject": {
                 "name":"backupfile.txt",
                 "path":"c:\\backup"
               },
       "type":"INCREMENTAL"
     }

request = json.dumps(backup)

# Issue the POST request to obtain a backup of the file by passing in the JSON
# serialized request as payload body.

c.request("POST", "/tsmcms/backupSystems/nodename/localObjects”,
  body=request, headers=headers)

r = c.getresponse()

j = json.loads(r.read().decode('utf-8'))

print (j);
```

2

The client management service REST API framework

Security considerations

For enhanced security, client management service complies with NIST SP-800-131A. Therefore, REST clients must ensure that they connect with TLS 1.2. There are no provisions today to support SSL or non-SSL connections.

The client management service protects all resources within the …/backupSystems/*nodeName* hierarchy by challenging for user name and password credentials, and authenticating to the Tivoli Storage Manager server. The credentials are passed in by using the traditional basic authorization mechanism (see Getting Started — A quick tutorial on using client management service).

Response and request message bodies

Response and request messages bodies adhere to the following principles whenever possible:

- Message bodies are formatted with JSON in the UTF-8 character set.
- References to resources are made with URLs instead of identifiers to reduce the work that is required by the client to generate resource URLs in successive requests.
- Date/time literal values use the following format (compatible with ISO-8601):

  `YYYY-MM-DDThh:mm:ssTZD`

  For example, 5 June 2015 4:23:45 PM in Pacific time zone is 2015-06-05T16:23:45-0700.

Paging requests

Some requests can respond with large data sets (more than 100K records). The client management service returns large data sets with multiple pages to reduce transmission time and avoid overrunning memory buffers.

The client can access a specific page by providing the query parameter **`pageNo`** (type is integer). If you do not specify the parameter, **`pageNo=0`** is the default behavior. The client management service can page forward quickly through large data sets, but paging backwards will be much slower. For this reason, read in a forward direction whenever possible.

The client can detect that a page of data was returned by looking for the **`nextPage`** property. If present, it contains a URL reference to the next page in the data stream.

The following example illustrates how you can use the **`pageNo`** query parameter to fetch page 2. The data is returned with the **`nextPage`** property set to indicate that more data is available and what URL to fetch.

```
GET https://localhost:9028/tsmcms/backupSystems/testNode/logRecords?pageNo=2

{
 << Log Records ...>>

 "nextPage":https://localhost:9028/tsmcms/backupSystems/TESTNODE/
       logs?pageNo=3
}
```

Filtering log records

Your REST client can use a filter on the GET request to return log records after a specific date. For example, you can obtain log records after 2015/02/02 at 1:23:34 PM in the Pacific time zone by issuing the following URL:

```
https://localhost:9028/tsmcms/backupSystems/TESTNODE/logRecords?
  filter=2014-02-05T13:23:34-0800
```

Long-running tasks

Some actions, like backup and restore, can take a long time to run. It is not practical to use a single synchronous REST call (such as POST) to handle this situation. The caller needs to know that the task was started and periodically poll the task status to determine when it is done.

When the task is initiated, client management service responds with a task object containing the task reference (called `taskRef`). The task reference points to the task resource that is used in successive GET requests to obtain status on the running task.

The HTTP return code for a long-running task is `Accepted` (202) instead of `Created` (201) or `OK` (200), which indicates that the request was accepted but not completed. In addition, client management service returns the URL to the task status that is located in the `Location` header.

The following example shows a sample transaction. The response includes a location header that points to the status resource in addition to the embedded task object with `taskRef` property.

Tip: For readability in the following example, the URL is not encoded. Ensure that you encode them before you send it so that problems with incorrect characters in the date/time string do not occur.

```
Request:
{
 "backupObject": {
  "name": "proposal.docx",
  "path": "c:\backup test\Current"
 },
 "type": "FULL"
}

Response:

Headers:
  Location = https://localhost:9028/tsmcms/backupSystems/TESTNODE/
        tasks/3871a68c-b68a-44cf-bdc6-3728a487957c

Payload:
{
 "backupTask": {
  "progress": 0,
  "started": "2015-06-11T10:31:43-0700",
  "state": "IN_PROGRESS",
  "taskRef": "https://localhost:9028/tsmcms/backupSystems
        /TESTNODE/tasks/
        3871a68c-b68a-44cf-bdc6-3728a487957c",
  "type": "BACKUP"
 }
}
```

Error response

If an error occurs, the HTTP code is set to the appropriate error code, and the body of the response is formatted as shown in the following example:

```
Error HTTP_CODE: {"httpStatus": HTTP_CODE,
"errorCode": RC, "errorString": MSG}
```

where:

HTTP_CODE is the HTTP Error code (for example, 200, 401, and so on).
RC is a client management service error code (see the following table)
MSG is auxiliary information that is related to the client management service error code

RC	Description
-1	Unknown Error
1	OK (No Error)
2	Client Not Registered
3	Node Not Configured
4	Authentication Failed
5	Server Communication Failed
6	Not Authorized
7	No Connection
8	Execution Error (unable to run command)

The following example shows an authentication failure where the user attempted to log in by using the user ID:

```
Error 401: {"httpStatus":401,"errorCode":4,"errorString":"foo"}
```

Client management service metadata for clients

You can access metadata about the client management service itself to ensure compatibility between your REST client and the client management service.

You can use the following URL resources:

`/tsmcms/about`: This resource shows the client management service version and list of configured backup systems (node names) without requiring the client to provide authentication credentials. Consumers can use this resource to "ping" the service.

`/tsmcms/backupSystems`: This resource provides the list of backup systems (node names).

`/tsmcms/backupSystems/`*`id`*: This resource shows the list of client management service capabilities that are enabled for a specified backup system named *`id`*.

Only the `LOG_QUERY` capability is enabled by default. All other capabilities need the user to run CmsConfig enable command to configure.

Extensions

You can incorporate your own resources into the client management service by configuring extensions. They make scripts or programs available as additional REST API resources.

To use extensions, follow these steps:

Step 1: Create the directory structure where the top-most directory has subdirectories named `get, put, post`, and `delete.` Within those subdirectories, you place your scripts or programs. These scripts or programs need to read and write JSON.

For example, to create a REST resource named “test” that can respond to GET, POST, PUT and DELETE requests, create the following directory structure:

```
SomeScriptDirectory
  get
    test
  post
    test
  put
    test
  delete
    test
```

All of the "test” programs write their response payloads as JSON and the `post/test` and `put/test` programs will also read JSON to get the request payload.

Each version of the `test` routine can do different things depending on the subdirectory it are in; for example, the `test` routine in the `get` subdirectory issues a query, whereas the `test` routine found in the `delete` subdirectory removes a resource.

Step 2: Enable and configure client management service. Issue the following commands:

```
CmsConfig setscriptdir nodeName scriptPath
CmsConfig enable nodeName SCRIPT_RUN
```

The `setscriptdir` command exposes the executable routine as a sub-resource under the extensions resource. The name of the executable routine is used as the name for the subresource.

For example, if you run the following request, the result is the output of the `test` executable routine that is in the `SomeScriptDirectory\get` directory. As you would expect, if you run a POST request, it will run the `post\test` executable routine, which passes in the request payload into the standard input, and so forth.

```
GET https://localhost:9028/backupSystems/TESTNODE/extensions/test
```

Limitations and known issues

This version of client management service has the following limitations and known issues:

- The extensions interface is synchronous and might time out if the script runs longer than a few minutes. The extensions resource is meant for simple, quick operations and not intended to be used for long duration operations. If you have to run a long-running command, run the command as a background process.

- The `dsmc` command must be in the system path for backup and restore functions to work.

- The backup and restore interfaces (`savedObjects` and `localObjects` resources) support globalized characters when the operating system is properly configured. A Microsoft Windows computer that is configured for English locale might not be able to work with double-byte character set (DBCS) (for example, Chinese characters). These non-displayable characters will be shown as a question mark (?) characters.

3

REST interface reference

By default, the Tivoli® Storage Manager client management service collects information only from client log files. To initiate other client actions, you can access the Representational State Transfer (REST) application program interface (API) that is included with the client management service.

- You can create REST applications to extend the capabilities of the client management service with scripts.
- Query and update client options files (for example, the `dsm.sys` file on Linux clients and the `dsm.opt` file on Linux and Windows clients).
- Query the status of the Tivoli Storage Manager client acceptor (CAD) and the scheduler.
- Back up and restore files for a client node.

about

`/tsmcms/about`

GET: Returns basic information about the client management service provider.

Tip: this request does not authenticate with the Tivoli Storage Manager server to enable discovery of services.

Response:

Property name	Value type	Description
`aboutRecord`	`AboutRecord`	See the following `AboutRecord` property table for details on the properties.

AboutRecord property table:

Property name	Value type	Description
version	String	The major and minor version of client management service. The major number identifies basic compatibility between your REST client and client management service, whereas the minor number tracks small changes. **Tip:** The version number does not align with product versions. It is only used to determine compatibility with client management service.
title	String	English text with the client management service name and version.
backupSystems	List<BackupSystem>	See the BackupSystem property table for a list of properties. The backupSystems property contains a list of backup systems that are configured in this client management service instance. You can obtain the same information by querying the root of the backup systems (for example, /backupSystems).

Example:

```
GET https://localhost:9028/tsmcms/about
```

```
{
 "aboutRecord":
  {
   "backupSystems":
    {
     "identifier":"TESTNODE",
     "backupSystemRef":"https://localhost:9028/tsmcs/backupSystems/TESTNODE"
    }
   ],
   "title":"Client Management Service Provider V1.2",
   "version":"1.2"
}
```

backupSystems

/tsmcms/backupSystems

GET: Returns information about all backup systems (for example, nodes, the Tivoli Storage Manager for Virtual Environments environment).

Tip: This request does not authenticate with the Tivoli Storage Manager server to enable discovery of services.

Response:

Property name	Value type	Description
backupSystems	List<BackupSystem>	See the BackupSystem property table for a list of properties.

Example:

```
GET https://localhost:9028/tsmcms/backupSystems

{
 "backupSystems": [
  {
   "backupSystemRef": "https://localhost:9028/tsmcms/backupSystems/ABC",
   "identifier": "ABC"
  },
  {
   "backupSystemRef": "https://localhost:9028/tsmcms/backupSystems/SAMPLE",
   "identifier": "SAMPLE"
  },
  {
   "backupSystemRef": "https://localhost:9028/tsmcms/backupSystems/TESTNODE",
   "identifier": "TESTNODE"
  }
 ]
}
```

/tsmcms/backupSystems/*ID*

GET: Returns information about a named backup system (for example, a node).

Response:

Property name	Value type	Description
backupSystem	BackupSystem	See the BackupSystem property table for a list of properties.

BackupSystem property table:

Property name	Value type	Description
identifier	String	A unique name for this backup system, for example, the node name. The identifier is always the same as the ***ID*** path parameter.
backupSystemRef	URL	A reference to the backup system resource that is referenced by this object.
capabilities	List<String>	The capabilities that are provided by this backup system. See the following table for a list of available capabilities.

The following table lists the set of capabilities that are provided by the backup system:

Capability name	Description
LOG_QUERY	Query for error logs for the backup system.
SCRIPT_QUERY	Query for the available scripts.
SCRIPT_RUN	Execute the given script.
CONFIG_QUERY	Query for configuration files.
CONFIG_UPDATE	Change configuration files.
DAEMON_QUERY	Query for daemon status (such as CAD or scheduler daemon).
DAEMON_UPDATE	Change daemon status (start, stop, or restart the CAD or scheduler).
FILE_BACKUP	Query and backup files from local storage.
FILE_RESTORE	Query and restore files from Tivoli Storage Manager.

Example:

```
GET https://localhost:9028/tsmcms/backupSystems/TESTNODE
```

```
{
 "backupSystem":
  {
   "identifier":"TESTNODE",
   "backupSystemRef": "https://localhost:9028/tsmcms/backupSystems/TESTNODE",
   "capabilities": [
    "FILE_QUERY",
    "FILE_BACKUP",
    "FILE_RESTORE",
    "LOG_QUERY",
    "CONFIG_QUERY"
   ]
  }
}
```

daemon

`/tsmcms/backupSystems/`*`ID`*`/daemon`

GET: Returns status of all CADs or scheduler services for a node.

Response:

Property name	Value type	Description
daemon	List<DaemonRecord>	See the DaemonRecord property table or a list of properties.

DaemonRecord property table:

Property name	Value yype	Description
name	String	The name of the service
runningState	String	The status for the services running backups for this node. Options are: STARTED, STOPPED, NOT_CONFIGURED
daemonRef	URL	A URL reference that points to the daemon service.

/tsmcms/backupSystems/*ID*/daemon/*serviceID*

GET: Returns status of a CAD or scheduler service for a node name.

Response:

Property name	Value type	Description
daemon	DaemonRecord	See the DaemonRecord property table for a list of properties.

PUT: Change the running state of a specific CAD or scheduler service for a node name. The service is restarted if the caller issues a PUT request with running state set to STARTED, and it was already running.

Request:

Property name	Value type	Description
daemon	DaemonRecord	See the DaemonRecord property table for a list of properties.

Response: (Reflects updated status)

Property name	Value type	Description
daemon	DaemonRecord	See the DaemonRecord property table for a list of properties.

Example:

```
GET https://localhost:9028/tsmcms/backupSystems/TESTNODE/daemon

{
 "daemon": [
  {
   "name":"TSM Backup Acceptor",
   "runningState":"STARTED",
   "daemonRef":"https://localhost:9028/tsmcms/backupSystems/TESTNODE/daemon/
            TSM Backup Acceptor"
  },
  {
   "name":"TSM Backup Scheduler",
   "runningState":"STARTED",
   "daemonRef":"https://localhost:9028/tsmcms/backupSystems/TESTNODE/daemon/
            TSM Backup Scheduler"
  }
 ]
}

GET https://localhost:9028/tsmcms/backupSystems/TESTNODE/daemon/TSM Backup Acceptor

{
 "daemon": {
  "name":"TSM Backup Acceptor",
  "runningState":"STARTED",
  "daemonRef": "https://localhost:9028/tsmcms/backupSystems/TESTNODE/daemon/
            TSM Backup Acceptor"
 }
}
```

extensions

`/tsmcms/backupSystems/ID/extensions`

GET: Returns a list of extension resource names.

Response:

Property name	Value type	Description
extensions	List<String>	A list of extension resources that are available.

Example:

```
GET https://localhost:9028/tsmcms/backupSystems/TESTNODE/extensions
```

```
{
 "extensions": [
  "status",
  "start",
  "stop",
  "reset",
  "cleanup"
 ]
}
```

`/tsmcms/backupSystems/ID/extensions/name`

GET: Runs the extension resource GET script and returns the output as payload from the request.

PUT: Runs the extension resource PUT script, which pipes the request payload into the input. The output is returned as a response.

POST: Runs the extension resource POST script, which pipes the request payload into the input and the output is returned as response payload.

DELETE: Runs the extension resource DELETE script, which returns the output as the response message.

Response Payload:

The output of the extension resource script. This output must be formatted in JSON.

If the extension program's return code is non-zero, the script returns a JSON payload of the following format:

```
{
 "httpStatus": <HTTP code>,
 "errorCode": <Process Return Code>,
 "errorString": <Error String>
}
```

Example:

```
GET https://localhost:9028/tsmcms/backupSystems/TESTNODE/extensions/cleanup

{
 "cleanupResponse": {
  "events": [
   {
    "2014-03-10T12:23:32-800", "Initialize System Variables"
   },
   {
    "2014-03-10T12:23:35-800", "Running Cleanup Phase"
   },
   {
    "2014-03-10T12:23:45-800", "Warning: Permission Denied"},
   {
    "2014-03-10T12:25:23-800", "Cleanup completed"
   }
  ],
  "status": "WARNING"
 }
}

GET https://localhost:9028/tsmcms/backupSystems/TESTNODE/extensions/bad_script

{
 "httpStatus":401,
 "errorCode":8,
 "errorString":"python: can't open file."
}
```

tasks

`/tsmcms/backupSystems/`*`ID`*`/tasks`

GET: Return the current status information for running or completed tasks. Only those tasks created by the user (as identified by using basic authentication) are returned.

Response:

Property name	Value type	Description
tasks	List<TaskRecord>	See the TaskRecord property table for a list of properties.

Example:

```
GET https://localhost:9028/tsmcms/backupSystems/TESTNODE/tasks

{
 "tasks": [
  {
   "progress": 0,
   "started": "2015-05-22T07:55:24-0700",
   "state": "IN_PROGRESS",
   "taskRef":"https://localhost:9028/tsmcms/backupSystems/TESTNODE/
          tasks/43a9a463-24bf-4538-a480-15d6c58c8397",
   "type": "BACKUP"
  },
  {
   "progress": 100,
   "started": "2015-05-22T07:55:23-0700",

   "finished": "2015-05-22T08:13:34-0700",
   "state": "OK",
   "taskRef": "https://localhost:9028/tsmcms/backupSystems/TESTNODE/
           tasks/fd6c0fff-20fd-4961-8ef3-6b649d8e8a4d",
   "type": "BACKUP"
  }

 ]
}
```

/tsmcms/backupSystems/*ID*/tasks/*taskID*

GET: Return the current status information for a running or completed task. Only those tasks created by the user are returned.

Response:

Property name	Value type	Description
task	TaskRecord	See the TaskRecord property table for a list of properties.

TaskRecord property table:

Property Name	Value Type	Description
type	String	The type of task with one of the following states: BACKUP, RESTORE
state	String	The status of the task with one of the following states: IN_PROGRESS, CANCELLED, FAILED, WARNING, OK
started	DateTime	The date and time that the task began.
finished	DateTime	The date and time that the task completed (null if still running).
progress	Integer	The progress of the task represented as a percentage (in the range 0 - 100).
taskRef	URL	A URL that points to the task record.

Example:

```
GET https://localhost:9028/tsmcms/backupSystems/TESTNODE/
     tasks/fd6c0fff-20fd-4961-8ef3-6b649d8e8a4d

{
 "task": {
  "progress": 100,
  "started": "2015-05-22T07:55:23-0700",
  "finished": "2015-05-22T08:13:34-0700",
  "state": "OK",
  "taskRef": "https://localhost:9028/tsmcms/backupSystems/TESTNODE/
           tasks/fd6c0fff-20fd-4961-8ef3-6b649d8e8a4d",
  "type": "BACKUP"
 }
}
```

localObjects

`/tsmcms/backupSystems/`*`ID`*`/localObjects[?path=`*`filePath`*`]`

GET: Returns a list of files and directories on the backup system. The `path` query parameter specifies the file system path to look into. If the path query parameter is missing, then the response will contain the top-level containers.

Response:

Property Name	Value Type	Description
`localObjects`	`List<ObjectRecord>`	See the ObjectRecord property table for a list of properties.

Example:

```
GET https://localhost:9028/tsmcms/backupSystems/TESTNODE/localObjects?
     path=C:\backup_test

{
 "localObjects": [
   {
    "created": "2015-02-13T07:56:43-0800",
    "modified": "2015-02-13T08:24:25-0800",
    "name": "Current",
    "path": "C:\backup test",
    "size": 4096,
    "type": "DIRECTORY"
   }
 ]
}
```

POST: The client can back up files by issuing a POST request against the `localObjects` resource by using the following request payload.

BackupRequest property table:

Property name	Value type	Description
backupObject	ObjectRecord	Specifies the objects to be backed up. When the backupObject property is missing (or null), the default backup action will be taken.
type	String	The type of backup to perform: INCREMENTAL or FULL. If the type property is not present, this value defaults to INCREMENTAL.

ObjectRecord property table:

Property name	Value type	Description
name	String	The file or directory name.
path	String	The file path to the parent of this object. The path uses the backup system's native file separator character. Windows uses backslashes and Linux uses forward slashes.
size	Long	The size of the object in bytes. This is null for non-file objects.
type	String	The type is one of FILESPACE, DIRECTORY, or FILE.
created	DateTime	The date and time when the file was created.
modified	DateTime	The date and time when the file was last modified.

Response:

Property name	Value type	Description
backupTask	TaskRecord	See the TaskRecord property table for a list of properties.

Notes:

- The response code for the POST request is Accepted (202)
- The Location HTTP header is set to the URL for the task status.

Example:

```
POST https://localhost:9028/tsmcms/backupSystems/TESTNODE/localObjects

Request Message:
{
 "backupObject": {
  "name": "proposal.docx",
  "path": "C:\backup_test\Current"
 },
 "type": "INCREMENTAL"
}

Response Message:
{
 "backupTask": {
  "progress": 0,
  "started": "2015-05-21T09:06:23-0700",
  "state": "IN_PROGRESS",
  "taskRef": "https://localhost:9028/tsmcms/backupSystems/TESTNODE/
          tasks/f9bc7d97-0620-4c65-b60e-52cae165c7c9",
  "type": "BACKUP"
 }
}
```

savedObjects

`/tsmcms/backupSystem/ID/savedObjects[?path=filePath]`

GET: Returns a list of saved files and directories that are located by the `path` query parameter. If the `path` query parameter is omitted, the list shows the top-level containers.

Response:

Property name	Value type	Description
savedObjects	List<ObjectRecord>	See the ObjectRecord property table for a list of properties.

ObjectRecord property table (for GET):

Property name	Value type	Description
name	String	The file or directory name
path	String	The file path to the parent of this object using native file separator character (for example, a backslash on Windows, and a forward slash on UNIX).
size	Long	The size of the object in bytes.
type	String	The type property is one of the following: `FILESPACE, DIRECTORY, FILE, VM, MAIL_SERVER, MAILBOX,` or `MAIL.`
created	DateTime	The date and time when the file was created.
modified	DateTime	The date and time when the file was last modified.
backupDate	DateTime	The date and time when the file was backed up.

Example:

```
GET https://localhost:9028/tsmcms/backupSystems/TESTNODE/savedObjects?
   path=\\c$\backup\Current

{
 "savedObjects":[
  {
   "path":"\\domain\c$\backup\Current\",
   "name":"bak",
   "type":"DIRECTORY",
   "created":"2015-02-13T08:59:53-0800",
   "modified":"2015-02-13T09:00:19-0800",
   "backupDate":"2015-04-09T15:49:41-0700",
   "size":0
  },
  {
   "path":"\\domain\c$\backup\Current\",
   "name":"proposal.docx",
   "type":"FILE",
   "created":"2015-02-13T08:57:07-0800",
   "modified":"2015-02-13T08:59:48-0800",
   "backupDate":"2015-04-09T15:50:02-0700",
   "size":11352
  }
 ]
}
```

POST: The client can restore files by issuing a POST request on the `savedObjects` resource by using the following payload.

RestoreRequest:

Property name	Value type	Description
restoreObject	ObjectRecord	See the ObjectRecord property table for a list of properties.
replace	Boolean	Indicates whether the restored files should replace existing files (if any). If the replace property is not present, files do not replace existing files.
destination	String	Identifies the path to restore files to. If this value is not present, the files are restored to original location.

ObjectRecord property table (for POST):

Property Name	Value Type	Description
name	String	The file or directory name. If this is null, the path including all of its subdirectories are restored.
path	String	The file path to the parent of this object using native file separator character (for example, a backslash for Windows, and a forward slash for UNIX).
backupDate	DateTime	The date and time to restore the object.

Response:

Property Name	Value Type	Description
restoreTask	TaskRecord	See the TaskRecord in the Tasks resource.

Notes:

- The response code is Accepted (202)
- The location HTTP header is set to the URL for the task status.

Example:

```
POST https://localhost:9028/tsmcms/backupSystems/TESTNODE/savedObjects

Request Message:
{
 "destination": "c:\tmp\restore1",
 "replace": false,
 "restoreObject": {
  "path": "c:\backup test\Current"
 }
}

Response Message:
{
 "restoreTask": {
  "progress": 0,
  "started": "2015-05-21T09:08:42-0700",
  "state": "IN_PROGRESS",
  "taskRef": "https://localhost:9028/tsmcms/backupSystems/TESTNODE/
           tasks/43b68fd3-1fdf-44db-b5a2-256ebf8a41be",
  "type": "RESTORE"
```

```
  }
}
```

configs:

`/tsmcms/backupSystems/ID/configs`

GET: Returns a list of configuration records for the backup system.

Response:

Property name	Value type	Description
configs	List<ConfigRecord>	See the ConfigRecord property table for a list of properties.

ConfigRecord property table:

Property name	Value type	Description
configFile	String	The path to the configuration file.
lines	List<String>	The configuration file contents represented as an array of strings.

Example:

```
GET https://localhost:9028/tsmcms/backupSystems/TESTNODE/configs

{
 "configs":[
  {
   "configFile":"c:\Program Files\Tivoli\TSM\baclient\dsm.opt",
   "lines":[
    "NODENAME TESTNODE ...",
    "PASSWORDACCSS GENERATE"
   ]
  }
 ]
}
```

PUT: Updates the configuration files. The payload is the same structure as the response.

Response:

Successful updates result in HTTP Response Code 204 (No Content) with no payload. Error codes are returned as usual.

Example:

```
PUT https://localhost:9028/tsmcms/backupSystems/TESTNODE/configs

Request Message:
{
 "configs":[
  {
  "configFile":"c:\Program Files\Tivoli\TSM\baclient\dsm.opt",
  "lines":[
    "NODENAME TESTNODE...",
    "# COMMENTED OUT -- PASSWORDACCSS GENERATE"
   ]
  }
```

```
 ]
}
```

Response: `204 (No Content)`

logRecords:

`/tsmcms/backupSystems/`*`ID`*`/logRecords[?pageNo=`*`page`*`&filter=`*`timeDate`*`]`

GET: Returns log records in sorted order from client management service. The **`filter`** query parameter is used to select a date to filter log records; only records after the specified date are returned. The **`pageNo`** query parameter is used to page through the log records (10,000 log records per page).

Response:

Property name	Value type	Description
logRecord	List<LogRecord>	See the LogRecord record for a list of properties.
nextPage	URL	If present, this is a complete URL to the next page in the sequence.

LogRecord is:

Property name	Value name	Description
logFile	String	The full path to the log file.
date	DateTime	The date and time of the log record.
subject	String	A unique identifier for the subject of the log message.
description	String	The message text itself in the original language.

Example:

```
GET https://localhost:9028/tsmcms/backupSystems/TESTNODE/logRecords?
     pageNo=0&filter=2014-08-28T8:17:40-0700
{
 "logRecords": [
  {
   "logfile":"c:\Program Files\Tivoli\TSM\baclient\dsmerror.log",
   "date":"2015-02-02T07:03:42-0800",
   "subject":"ANS1577I",
   "description":"The Windows console event handler received a
           'Close' console event."
  },
  {
   "logfile":"c:\Program Files\Tivoli\TSM\baclient\dsmerror.log",
   "date":"2015-02-02T07:24:32-0800",
   "subject":"ANS0361I",
   "description":"DIAG: sessRecvVerb(): Invalid verb received."
  },

  ...

 ],
 "nextPage":"https://localhost:9028/tsmcms/resources/TESTNODE/logRecords?
         pageNo=1&filter=2014-08-28T8:17:40-0700"
}
```

CmsConfig commands

Use the CmsConfig command to enable capabilities in the client management services for REST API.

CmsConfig enable command

Use the **CmsConfig enable** command to configure the Tivoli® Storage Manager client management service to collect other diagnostic information from a backup-archive client or to initiate actions on the backup-archive client.

By default, the client management service collects only client log files. You can access the REST API that is included with the client management service to initiate other client actions. With the **CmsConfig enable** command, you can configure which REST API feature is enabled from the client management service.

Syntax

```
CmsConfig enable nodeName capability
```

Parameters

nodeName
The client node name that is associated with the log files. For most client systems, only one node name is registered to the Tivoli Storage Manager server. However, on systems with multiple users, such as Linux client systems, more than one client node name can be used. This parameter is required.

capability
The following list summarizes the capabilities that you can enable with the client management service REST API. This parameter is required.

LOG_QUERY
Reads log records from the client management service. This capability is enabled by default.

CONFIG_QUERY
Reads the options files for a client node (for example, the `dsm.opt` file and the `dsm.sys` file on Linux clients).

CONFIG_UPDATE
Updates the options file for a client node.

DAEMON_QUERY
Queries the status of the client acceptor, scheduler, or both.
For example, this capability queries whether these services are running, not running, or not installed.

FILE_BACKUP
Initiates a backup operation on a file. This capability implicitly allows your REST client to query for local file information.

FILE_RESTORE
Initiates a restore operation of a file that was backed up to the Tivoli Storage Manager server. This capability implicitly allows your REST client to query for file information that is saved on the Tivoli Storage Manager server.

SCRIPT_RUN
Extends the client management service with scripts.

Example for a Linux client system

To enable the capability to initiate a backup operation on a file on the node SUSAN, issue the following command from the `/opt/tivoli/tsm/cms/bin` directory of the client system:

```
./CmsConfig.sh enable SUSAN FILE_BACKUP
```

Example for a Windows client system

To enable the capability to initiate a backup operation on a file on the node SUSAN, issue the following command from the `C:\Program Files\Tivoli\TSM\cms\bin` directory of the client system:

```
CmsConfig enable SUSAN FILE_BACKUP
```

CmsConfig disable command

Use the **`CmsConfig disable`** command to remove REST API features that you enabled from the Tivoli Storage Manager client management service.

Syntax

`CmsConfig disable` *`nodeName capability`*

Parameters

`nodeName`
The client node name that is associated with the log files. For most client systems, only one node name is registered to the Tivoli Storage Manager server. However, on systems with multiple users, such as Linux client systems, more than one client node name can be used. This parameter is required.

`capability`
The following list summarizes the REST API features that you can disable from the client management service.

LOG_QUERY
Reads log records from the client management service. This capability is enabled by default.
CONFIG_QUERY
Reads the options files for a client node.
CONFIG_UPDATE
Updates the options file for a client node.
DAEMON_QUERY
Queries the status of the client acceptor, scheduler, or both.
FILE_BACKUP
Initiates a backup operation on a file.
FILE_RESTORE
Initiates a restore operation of a file that was backed up to the Tivoli Storage Manager server.

SCRIPT_RUN
Extends the client management service with scripts.

Example for a Linux client system

To disable the capability to initiate a backup operation on a file on the node SUSAN, issue the following command from the `/opt/tivoli/tsm/cms/bin` directory of the client system:

```
./CmsConfig.sh disable SUSAN FILE_BACKUP
```

Example for a Windows client system

To disable the capability to initiate a backup operation on a file on the node SUSAN, issue the following command from the `C:\Program Files\Tivoli\TSM\cms\bin` directory of the client system:

```
CmsConfig disable SUSAN FILE_BACKUP
```

CmsConfig setscriptdir command

Use the **`CmsConfig setscriptdir`** command to specify the location where the scripts extending client management service REST API are run. Use this command if you enabled the `SCRIPT_RUN` capability with the **`CmsConfig enable`** command.

Syntax

`CmsConfig setscriptdir` *`nodeName scriptDir`*

Parameters

`nodeName`
The client node name. For most client systems, only one node name is registered to the Tivoli Storage Manager server. However, on systems with multiple users, such as Linux client systems, more than one client node name can be used. This parameter is required.

`scriptDir`
The directory path where the scripts are located. This parameter is required.

Example for a Linux client system

Specify the script directory as `/opt/tivoli/tsm/cms_scripts` for the node SUSAN. From the `/opt/tivoli/tsm/cms/bin` directory, issue the following command:

```
./CmsConfig setscriptdir SUSAN /opt/tivoli/tsm/cms_scripts
```

Example for a Windows client system

Specify the script directory as `C:\Program Files\Tivoli\TSM\cms_scripts` for the node SUSAN. From the `C:\Program Files\Tivoli\TSM\cms\bin` directory, issue the following command:

```
CmsConfig setscriptdir SUSAN C:\"Program Files"\Tivoli\TSM\cms_scripts
```

Notices

This information was developed for products and services offered in the U.S.A.

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

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

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

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

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

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PAPER “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

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

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

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758
U.S.A.

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

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

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems.

Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

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

COPYRIGHT LICENSE:

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

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, ibm.com, [insert IBM brand that materials relate to ONLY IF it appears on our trademark Web site], and [insert IBM product name that materials relate to ONLY IF it appears on our trademark Web site] are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

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

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