TADDM

rest_in_ease

REST interaction made easy

User's Guide and Reference

Morten Moeller
WW Technical Evangelist
Tivoli Software, IBM SWG
version 1.0.2
Date printed: March 8, 2013

Copyright Notice

Trademarks

Notices

Preface

Since version 7.2, TADDM has provided a Representational state transfer (REST) API interface through which users can perform updates to the TADDM database. The interface allows users to programmatically access the TADDM database to query and update resource information, and even create new resources.

The REST interface is intended to be used to update resource configurations without having to perform a TADDM scan of the resources. This allows programs – for example IBM Tivoli Monitoring and IBM Tivoli Monitoring for Virtual Servers monitoring agents – to update resource configuration and relationship information in the TADDM database in real time. For example, when a VMware hypervisor changes the memory configuration of a server, the monitoring agents will be notified, and they can in turn update the information in the TADDM in order to reflect the changes that were implemented. As a matter of fact, this behavior is provided out-of-the-box in the latest versions of the IBM Tivoli Monitoring agents – but you can also use the REST interface to implement similar functionality to your custom agents.

You can also use the REST interface to create a user-friendly command-line based interface to TADDM that for example can be used to update trivial attributes such as assetTag, locationTag or administrator of multiple resources in a single operation.

Prior to TADDM version 7.2.1, the changes applied to the system through the REST interface - also known as *proactive* changes – would not be processed by the topology and change manager components until the next execution of a discovery. This means, that there were no way – except for scheduling a reoccurring discovery – to control when the proactive changes would be processed. With the introduction of transactional processing in TADDM 7.2.1 this has all changed. In TADDM 7.2.1 the topology builder and change manager tasks execute, along with several other system tasks, based on a schedule. During this process, changes – including proactive changes – will be processed, and if the system has been configured to do so, notifications related to the changes can be forwarded to the operational staff…… all without a single discovery. Option you have when the change manger has discovered changes are for example to initiate an ITIC upload to CCMDB, or a rediscovery or a full discovery of the proactively changed resources, for example after the reception of 10 proactive changes.

As you understand, the combination of a REST enabled agent and the transactional processing opens up for previously unseen dynamics in the TADDM solution which ultimately will lead to even tighter and better control of your environment.

Table of Contents

1

1 rest_in_ease

The rest_in_ease utility has been developed to allow you to easily list, create, and update TADDM resources from a command line.

The utility offers the following facilities:

- Select and identify resources based on one of more of:
  - A list of identifying attributes
  - Membership of a TADDM group
  - GUID
  - Resource type and optional identifying attributes
- Export all or a subset of attributes for a selection of resources
- List selected attributes based on a selection of resources.
- Update common attributes for a selection of resources.
- Create new resources based on a list of attributes.
- Modify - update or create - resources based on a list of attributes.

1.1 Prerequisites

Rest_in_ease is based on Jython 2.5.2, which must be installed on the system from which you want to run rest_in_ease. Jython 2.5.2 can be downloaded and installed from http://mac.softpedia.com/get/Developer-Tools/Jython.shtml or from http://sourceforge.net/projects/jython/files/latest/download.

Installation instructions for jython2.5.2 are provided on the site, but for all platforms, you simply set the JAVA_HOME environment variable, and run the installation using these sample commands:

Windows:

```
set JAVA_HOME=<your jre path>
set PATH=%PATH%;%JAVA_HOME%\bin
java -jar <your java download location>\jython_installer-2.5.2.jar
```

Your JAVA_HOME will most likely be similar to C:\Program Files (x86)\IBM\Java60\jre.

Unix:

```
export JAVA_HOME=<your jre path>
export PATH=$PATH:$JAVA_HOME/bin
java -jar <your java download location>/jython_installer-2.5.2.jar
```

Your JAVA_HOME will most likely be similar to /opt/ibm/java-x86_64-60/jre.

1.2 Installation and customization

To install rest_in_ease, simply unpack the accompanying archive (zip) file to any directory on your workstation, customize it for your environment, and you are good to go.

Windows:

Complete these steps to install rest_in_ease in your Windows environment:

1 Unpack the rest_in_ease archive file to any location on your system.

2 Edit the rest_in_ease.bat file, and modify the JYTHON_HOME variable so it points to the location where you installed Jython V2.5.2.

A sample rest_in_ease.bat file looks like this:

```
@echo off
REM
REM    use this file to execute rest_in_ease
REM
REM
REM set CUR_DIR=%~dp0
SET JYTHON_HOME=C:\jython2.5.2
%JYTHON_HOME%\jython.bat rest_in_ease.py %*
```

3 To execute the rest_in_ease utility simply move to the installation directory, and issue this command:

```
rest_in_ease.bat <your arguments>
```

Unix

Complete these steps to install rest_in_ease in your Unix environment:

1 Unpack the rest_in_ease archive file to any location on your system.

2 Ensure that the files are unix-encoded by running this command:

```
cd <unpack_directory>
dos2unix rest_in_ease.*
```

3 Ensure that all users can execute the utility by issuing these commands:

```
cd <unpack_directory>
chmod +x rest_in_ease.*
```

4 Configure the runtime environment to support rest_in_ease:

In unix environments you have two options to invoke rest_in_ease. Either customize the rest_in_ease.sh file similar to the Windows approach, or add the jython installation directory to your PATH environment variable.

Using rest_in_ease.sh:

a Edit the rest_in_ease.sh file, and modify the JYTHON_HOME variable so it points to the directory in which you installed jython V2.5.2.

A sample rest_in_ease.sh file looks like this:

```
#!/bin/bash
#
#  Use this script to invoke rest_in_ease
#
JYTHON_HOME=/opt/jython2.5.2
${JYTHON_HOME}/bin/jython rest_in_ease.py $@
```

b To execute the rest_in_ease utility simply move to the installation directory, and issue this command:

```
rest_in_ease.bat <your arguments>
```

Adding jython to your PATH:

Instead of using the rest_in_ease.sh script, you can invoke the rest_in_ease.py script directly, if you ensure that the <jython_install>/bin directory is in your PATH environment variable.

a Modify our user profile (found in your HOME directory) and include the <jython_install>/bin directory in the path.

For example, add the fillowing lines to near the end of the .bash-profle or .profile file:

```
JYTHON_BIN=/opt/jython2.5.2/bin
export PATH=${PATH}:${JYTHON_BIN}
```

b To execute the rest_in_ease utility simply move to the installation directory, and issue this command:

```
rest_in_ease.py <your arguments>
```

Defining your runtime environment

rest_in_ease needs a few customization parameters in order to access your TADDM environment. Among these are the host name of the TADDM Domain, Enterprise, or Storage Server, and the credentials needed to access the TADDM environment. All of this information can be provided as invocation arguments, but you can also pre-configure them in the rest_in_ease.ini file or any other file of your choice. By default. rest_in_ease reads the rest_in_ease.ini file when it is started, but you can overwrite this behavior using the `-i <configuration-file>` argument. Details are provided in the remainder of this document.

To pre-configure the options necessary to connect to your TADDM environment, and optionally set default values for selected optional options such as fetchSize, queryDepth or reporting options simply provide your preferences in the rest_in_ease.ini file.

The default rest_in_ease.ini file looks like this:

```
##
## Set default options for rest_in_ease
##
##  TADDM HOST (hostname or IP address)
--host taddmPRI.tivoli.edu
##  TADDM Port
--port 9430
## TADDM user
--user administrator
## TADDM password
--password collation
## default action is set to list the resources
--action list
##  Limit the output to the first 10 lines
## ( use the  '-S' invocation argument to overwrite, or change
##   the current setting to '-- fetchsize 0' to see all records)
--fetchsize 10
## add as many preferred options here as you like
```

1.3 Syntax

The syntax of the command-line invocation is:

```
rest_in_ease.bat/sh  [COMMAND_OPTIONS] [TADDM_OPTIONS] [INPUT_OPTIONS]
[RESOURCE_OPTIONS] [QUERY_OPTIONS] [ATTRIBUTE_OPTIONS] [REPORT_OPTIONS]
[OUTPUT_OPTIONS] [MSSREGISTER_OPTIONS}
```

1.3.1 COMMAND_OPTIONS

These options are used to control the function, and interaction with rest_in_ease.

Syntax

```
-a|--action {list | attr | export | update | create | modify | delete}
[-f|--force] | -v|--version
```

Description

`action`	Specifies the action you want to perform. The names are self-explanatory, except for modify. This action will update existing resources, and create non-existing resources.
`-f|--force`	Used to avoid being prompted for confirmation to apply updates. This option is only enforced for the update, create, modify and delete actions.
`-v|--version`	Shows the current version of rest_in_ease.

Example

For example, the following:

```
-a list -S 40
```

will list 40 resources based on qualification provided in the RESOURCE_OPTIONS clause.

1.3.2 TADDM_OPTIONS

These options control access to the TADDM storage server.

Syntax

```
-H|--host <hostname> -P|--port <port> -u|--user <username>
-p|--password <password>
```

Description

`hostname`	The hostname or IP address of the TADDM storage server you will connect to.
`port`	The port number you want to use for the connection to the TADDM storage server. Default value is 9430.
`username`	Name of the user used to authenticate with the TADDM storage server.
`password`	Password for the user used to authenticate with the TADDM storage server.

1.3.3 INPUT_OPTIONS

These options are used to read command line arguments from a file.

If this argument is not provided, the utility will look for a file named *rest_in_ease.ini* in the same directory as the utility itself, and automatically use this file if it exists. If you want to use any other standard arguments stored ina file, use the *-i <file_name>* argument.

Syntax

```
-i|--input <file_name>
```

Description

<file_name>	File name according to the syntax of your operating system. The name specified is relative to the path where rest_in_ease resides.

Example

In the specified file, you can provide any of the other arguments, each on a single line.

For example, if you create a file named rest.input with the following content:

```
-H tadmserver.tivoli.com
-P 9430
-u administrator
-p collation
```

and provide the -i rest.input argument to the command line, rest_in_ease will read the arguments from the file. If you also provide any argument, except for the –i/--input, on the command line, the command line provided arguments will take preference. If you invoke rest_in_ease like this:

```
rest_in_ease –i rest.input –p mypw
```

you will use the value of mypw to authenticate with the TADDM storage server.

If you name your control file rest_in_ease.ini, you do not need to reference it when invoking rest_in_ease.py.

1.3.4 RESOURCE_OPTIONS

These options are used to select the resources to process. You can select resources based on a combination of attributes, membership of a TADDM group, specific GUIDs, or a MQL query.

Syntax

```
[LIST_OPTIONS] [GROUP_OPTIONS] [GUID_OPTIONS] [TYPE_OPTIONS]
-k|--combined
```

Description

`-k|--combined`	If used, this flag will be used to combine resources from multiple options, thereby allowing you to produce a list of resources from for example a list, a couple of guids, and specific resources identified by type and filter.

When rest_in_ease executes, resources are selected in the order in which they are provided. If you specify multiple resource options, only the first will be processed if the --combined is not specified. If it is, the resource options will be processed in the order in which they have been supplied.

The processing stops when the desired maximum number of resources (specified with the –S | --fetchsize arguments) have been reached.

1.3.5 LIST_OPTIONS

Use the list option to select resources from a list of attribute:value pairs.

Syntax

```
-L|--list <resourceType>:{<attribute:value>{,<attribute:value>}*} | <file_name>
```

Description

resourceType:	The class name of the resource you are identifying with the attribute:value pairs.
attribute:value	A json formatted list of attribute:value key-value pairs separated by commas. Values should be enclosed in single quotes, and the whole list must be enclosed in braces({}).
file_name	The file name, relative to the current directory, that contains json formatted attribute:value key-value pairs. Each line represents the attributes:value key-value pairs to identify a single resource.

Examples

The following attribute:value string can be used to provide the attributes necessary to identify one or more resources:

```
-L ComputerSystem:{assetTag:'Las Vegas',type:'IpDevice'}
```

To provide attribute:value pairs for multiple queries, separate them with a comma. The following can be used join two queries:

```
-L "ComputerSystem:{signature:'pete'}, ComputerSystem:{signature='joe'}"
```

To identify a unique resource based on attributes, you can provide enough attributes to satisfy at least one naming rule.

If you want to read the attribute:value pairs from a file named rest.list, you can use the following syntax. The list may be created using the export action.

```
-L rest.list
```

1.3.6 GROUP_OPTIONS

Use these options to select resources from TADDM groups: Collections, Business Applications, or Business Services. When using this option, rest_in_ease will select the members of the group down to the lowest level.

Syntax

```
-C|--groups {<GroupType:GroupName>{,<GroupType:GroupName>}* | <file_name>} [-m|--
members groups | resources]
```

Augments

GroupType	The CDM class name of a group that is supported by the TADDM Grouping Composer. One of the following: BusinessSystem, Application, Collection, AccessCollection
GroupName	Name of the group to. The GroupType:Group:Name value pairs bust be provided as a comma separated list. It is used to identify the groups you want to process. GroupNames should be enclosed in single quotes.
file_name	The file name, relative to the current directory that contains GroupType:GroupName key-value pairs. Each line represents the GroupType:GroupName key-value pairs to identify a single group.

Description

`-C|--groups`	A comma separated list of GroupTyoe:GroupName pairs, or a reference to a file that contains a list of GroupType:Group:Name pairs, which is used to identify the members of the group(s) so they can be processed in one interaction. This option is handy if a workload has been moved to a different site, and the monitoring agents need to update TADDM to reflect the new configuration.
`-m|--members`	Specifies which group resources to include in the report. You can select members or groups. If this argument is not specified, the report will show all group members.

Examples

Use the following syntax to see all the resources that members of the collection named ‘Pulse 2012’:

```
-C “Collection:’Pulse 2012’”
```

To see only the functional groups for the Business Service named Order Entry, use this format:

```
-C “BusinessSystem:’Order Entry’” –m groups
```

Use the following syntax to see all the resources without children (appServers, computerSystems, and empty functional groups) that are related to the business application named ‘Billing’:

```
-C “Application:Billing” –m members
```

To select resources from a file, for example named rest.groups, you can prepare a file with the content similar to the following:

```
BusinessSystem:’Order Entry’
Collection:’Pulse 2012’
Application:Billing
Application:’Logistics Management’
```

and invoke the rest_in_ease utility with the following option:

```
-C rest.groups
```

1.3.7 GUID_OPTIONS

Use the options to select resources based on a list of GUIDs.

Syntax

```
-G|--guids <guid>{,<guid>}* | <file_name>
```

Arguments

guid	A comma separated list of guids.
file_name	The file name, relative to the current directory that contains guids. Each line represents a single guid used to identify a resource.

Description

`-G|--guids`	A comma separated list of guids, or a reference to a file that contains a list of guids, which is used to identify resources to process.

Examples

To select a single resource based on guid, use the following syntax:

```
-G <guid>
```

To select multiple resources from a list of guids, prepare a file, for example named rest.guids, with content similar to the following:

```
{5AB8EDD0EB5039D189F40B369AEAFFF0}
{57B1F48031713D6F91CEC0D97554DAEB}
19127EC801673417AAFB212FA39D08CE
6FF5B335E30D3BFAB78353E644D081FA
840F0CB2F30C3FA288CA347D9BAB3D01
{FB15D41268003EACB597E7688C735B10}
```

Notice that the guids can be enclosed in braces.

Then, invoke rest_in_ease with the following option:

```
-G rest.guids
```

The file can be created using the export action in combination with the–g option and optionally the –o option.

1.3.8 TYPE_OPTIONS

Use the options to select resources based on a MQL query that searches for specific resources types that contain certain attribute values

Syntax

```
-T|--types <resourceType>{,<resourceType}* [-F|--filter <where_clause> | file_name]
[-M|--mss <mss_guid>|<mss_name>][-D|--disctinct]
```

Arguments

resourceType	Any valid resource type that is known in the CDM.
where_clause	A valid MQL where clause used to qualify the resources you select.
file_name	The name of a file, relative to the current directory, that contains the filter.
`mss_guid`	The guid of the management software system with which you will register the changes.
`mss_name`	The name of the management software system with which you will register the changes.

Description

`-T|--types`	A comma separated list of CDM class names of the resources to select – or of the resource types to include in a compound query
`-F|--filter`	Used to qualify the instances of the resource types you are selecting. The filter is used to build the where clause of a MQL query.
`-D|--distinct`	Instructs the rest_in_ease tools to select ONLY resources of the specified type. Similar to the ONLY clause in MQL queries. If multiple used resource types are specified, the filter will be applied to each one individually, thus allowing you to create a report that contains multiple resource types from single query. Naturally, all the resource types must support all the attributes used in the filter.

Examples

To find all types of computer systems, use the following options:

```
-T ComputerSystems
```

Remember that the number of records returned is limited by the –S option.

To find all types of computer systems, applications, and collections use the following options:

```
-T "ComputerSystems, Applications, Collections"
```

Some resource types, for example ComputerSystem, and Collection, contain sub-types that by default are included when you select these resource types. To select only resources of the specified type, apply the –D or --distinct option.

To select ONLY computer systems, use the following syntax:

```
-T "ComputerSystems" -D
```

To limit the selected resources to those that have specific attribute values, add the –F or –filter option. As in MQL, you can filter on any known attribute for the specified resource type, so if you apply multiple resource types, you must filter on attributes that are known to all the specified resource types – for example displayName, or assetTag.

To select any type of computer system which contains the string ‘abc’ in the displayName use the following syntax:

```
-T ComputerSystem –F “displayName contains ‘abc’”
```

Remember that you can use the attr action to list all the valid attributes for a given resource.

To find the results of a composite query, one that uses data from multiple object types, simply specify multiple resource types, define the correlation in the filter (-F|--filer) and do not use the –D|--distinct option. For example, to select all the computer systems that are associated with a site for which the name contains an ‘o’, you can use the following options:

```
-T "ComputerSystem, AdminInfo" -F "ComputerSystem.guid == AdminInfo.objGuid and
AdminInfo.site contains 'o'"
```

For complex filters, you can save them in a file, and provide the file name as the value for the –A argument.

1.3.9 QUERY_OPTIONS

These options are used to control how resources are being fetched from the TADDM database.

Syntax

```
[-S|--fetchsize <max resources>] [-Q|--depth <depth>]
```

Description

`max resources` — Determines the maximum number of records to process. Default value is 1.

You should modify this option with care. Processing too may records may impact performance.

This option does not apply if you use the –C|--groups option to list all descendants of a group object.

`depth` — This integer controls the level of details from related resources that are collected from the TADDM database for each resource. The higher the number the more details. Default value is 1.

Use this option with extreme care. Gathering a lot of information about related resources may seriously impact the performance, and without providing a lot of benefit.

Examples

While the majority of the report options should be self-explanatory, you should notice the –S option. This is used to control the number of records you process, and the default is 1.

To process more than a single resource, you should consider using the –S option to increase you ‘batch size’. To process 50 records, for example to update the locationTag, you can use these options:

```
-a update-T ComputerSystem –F “assseTag contains ‘32’” –A “locationTag:’Tokyo’” –S
50
```

1.3.10 ATTRIBUTE_OPTIONS

Use these options to:

1 Select attributes to be displayed in the report or to be exported

2 Provide new values for attributes to update, create of modify

You can use the same format for both read actions (list, attr, and export) as well as write actions (update, create, modify). If you do, the values provided will be ignored if you are using a read action.

Syntax

```
-A|--attributes <attribute>[:<new_value>] {,<attribute>[:<new_value>]}* } |
<file_name>
```

Arguments

`attribute`	Name of the attribute you want to see in the report, or manipulate using the create, update, or modify actions. Notice, the attribute must be valid for the resource type(s) that have been selected.
`new_value`	Required for create, update, and modify actions. Provides the new value for the attribute that you want to manipulate.
`file_name`	The file name, relative to the current directory, that contains json formatted attribute:new_value key-value pairs. Each line represents an attributes:new_value key-value pairs to identify a single resource.

Description

`-A|--attributes`	A comma separated list of attributes, and optional new values, which is used to control the display or modification of resource attributes. The new_value is only used to rest_in_ease performs modifications (create, update, modify), and are ignored for the list, attr, and export actions. Attribute:new_value pairs may be stored in a file which can be referenced instead of providing the attribute:new_value pairs on the command line.

Examples

To see the assetTag, locationTag, and signature attributes for computer systems use either of the following formats :

```
-A "assetTag, locationTag, signature"
-A "assetTag:'asset-23', locationTag:'Las Vegas', signature='Pulse 2012"
```

In case the action is list, attr or export, the values you have provided in the second option will be ignored.

To for example update the locationTag attribute for all the selected resources, use the following syntax:

```
-A "locationTag:'Las Vegas'"
```

To create a new resource, you MUST provide values for enough attributes to satisfy at least one naming rule. Naming rules are specific to each resource type. Refer to the CD documentation for details. To create a computer system and satisfy the CSProduct naming rule, you can use the following syntax:

```
-A "manufacturer:'IBM', model:'9877-88L', serialNumber:778866-983'"
```

Naturally you can add additional attributes to meet your needs.

If you are performing mass updates, or inserts, you can prepare a file that contains a set of attributes to use on each line. For example, to modify – update or create – a number of computer system resources to indicate in the lifecycleState that the system have been tested, you can prepare a file named `rest.modify` with content similar to the following:

```
{signature:192.168.100.11,lifecyclaState:5}
{signature:192.168.100.22,lifecyclaState:5}
{signature:192.168.100.33,lifecyclaState:5}
{signature:192.168.100.44,lifecyclaState:5}
```

Then, to apply the modifications, use the rest_in_ease tool with the following options:

```
-a modify –A rest.modify
```

This will update the lifecycleState attribute for all the known computer systems, and since you have provided attributes to support a naming rule, any non-existing computer system resources will be created – with a lifecycleStatus of 5.

1.3.11 REPORT_OPTIONS

Use the report options to control the content of the results that are produced by rest_in_ease. These options apply primarily to the list, attr and export options.

Syntax

```
[-c|--compact] [-g|--guid] [-n|--noIDs] [-O|--orderBy <orderBy>]
[-s|--shortFormat]
```

Arguments

`orderBy`	Name of the attribute you want to use to sort the list. Only ascending ordering on a single attribute is supported. The attribute must be one of the three default attributes (guid, _class, displayName) or specified using the –A or –attributes argument.

Description

`-c|--compact`	Prevents printing of attributes for which the value is null.
`-g|--guidOnly`	Forces the export action to export only the guids of the selected resources. Applies only to the export action.
`-n|--noIDs`	Strips attributes such as guid, lastModifiedTime, lastModifiedBy, and type from the export data so they can be used by the create, attr, or list actions without modification.
`-O|--orderBy <orderBy>`	Sorts the output in ascending order of the selected attribute. This option is only used for the list and attr actions.
`-s|--shortFormat`	Forces the class names of to be printed in the short format.

Examples

While the majority of the report options should be self-explanatory, you should notice that the –orderBy (-O) option only applies to the formatted report. When resources are fetched from the TADDM database, there is no guarantee, especially when using the –L, -C, and –G options, that resources will be fetched in the correct sequence. To provide a report of all Applications ordered by displayName, you can use the following arguments:

```
-a list –T Application –S 0 –O "displayName"
```

1.3.12 OUTPUT_OPTIONS

These options help you control the output provided by the rest_in_ease tool.

Syntax

```
[-B|--banner] [-l|--log <logFile>]  [-o|--out <outFile>]
[-e|--err <errFile>] [-d|--debug] [-t|--trace] [-h|-help] [-q|--quiet]
```

Arguments

`errFile`	File name to which error information is written. Default is <syserr>.
`logFile`	File name to which debug and trace information is written. Default is <sysout>.
`outFile`	File name to which the output from rest_in_ease is written. Default is <sysout>.

Description

`-B|--banner`	Show the input options
`-d|--debug`	Provides information helpful when debugging the tool.
`-e|--err`	Allows you to write the error messages to a file.
`-l|--log`	Allows you to write log information to a file.
`-h|--help`	Provides online help information.
`-o|--out`	Allows you to direct the rest_in_ease reports directly to a file. This option is mainly used in conjunction with the export action.
`-q|--quiet`	Suppresses console messages.
`-t|--trace`	Provides detailed information about the http calls performed by the tool. If used in combination with –d or –debug, this will also trace the data parsed from one routine to another.

1.3.13 REGISTER_OPTIONS

Use these options to register your updates with a specific Management Software System.

Syntax

```
[-R|--registerMss [<mss_guid]> | <mss_name>]
```

Arguments

`mss_guid`	The guid of the management software system with which you will register the changes.
`mss_name`	The name of the management software system with which you will register the changes.

Description

`-R|--registerMss`	Sets the identity of an existing Management Software System and ensures that the changes you apply (using the either the create, update, or modify action) will be registered with the correct management software system. If you provide an empty value for the --registers option, your updates will not be registered with a MSS. If the --registerMss option is not provided on invocation of tool, the updates will be registered with rest_in_ease's own MSS definition: *TADDM_Proactive_Update*.

Examples

To register updates with named MSS, use the following syntax:

```
-R "ibm-
cdm:\/\/\/CDMMSS\/Hostname=itm.tide.ibm.com+ManufacturerName=IBM+ProductName=IBM
Tivoli Monitoring Services"
```

As an alternative, it might be easier to use the guid. For example:

```
-R 27C1E9FB31123D2281E18D7D363E9E64
```

To avoid registration of the updates with a management software system, simply use this syntax:

```
-R ""
```

Remember that you can list all management software system using these options:

```
A list -T "ManagementSoftwareSystem"
```

Feel free to add a --filter argument to limit the results.

1.4 Sample files

To provide default values for selected arguments, for example credentials and tad host definitions, you can use the `-i` option, and provide the name of a file that contains your preferred settings.

The following shows a sample input file:

```
--action list
--banner
--password collation
--host mm13
--port 9430
--fetchsize 10
--user administrator
```

If you name this file rest.ini and place it in the directory from which you invoke the rest-in-ease utility, and provide the following argument `-i rest.ini` you will always apply the settings in the file.

1.5 Argument reference

The meaning of the various arguments that can be parsed to rest_in_ease are:

`-a|--action <action>`

Valid actions are:

`list` lists the guid, type, and displayName of the selected resources.

The resources to be listed are selected through the --list (-L), --groups (-C), --guids (-G), or --type (-T) arguments.

If you also specify the --attributes (-A) argument, the list will be expanded with the attribute names listed in this argument. Values provided in the attribute option are ignored.

You can also apply the –orderBy (-O) argument to sort the list in ascending order of the selected attribute.

`attr` lists all available attributes for the selected resources.

If you also provide the --attributes (-A) argument, the list will be limited to the specified attributes.

If you apply the --compact (-c) argument, only attributes that contains a value will be listed.

If you provide the shortformat option (-s) class names will be displayed in the short format.

`export` Exports the selected attributes to a flat file.

The --compact (-c), --shortformat (-s), and --attributes (-A) arguments control the details that are exported - similar to what applies to the attr action. In addition, you can use the --guid (-g) argument to export a simple list of GUIDs.

`update` Updates attributes for the selected resources

The --attribute (-A) argument is used to define values for attributes to be updated using the " `<attribute>:<value> {, <attribute>:<value>}*`" format

`create` Creates new resources in the TADDM database

To create new resources, you must specify the --list (-L), or --type (-T) arguments. If you use the --type argument the attributes and values must be specified using the –attributes (-A) argument, and you must specify enough attribute:value pairs to support at least one naming rule for the selected resource type.

To automatically create the resources, apply the --force (-f) argument.

`modify` Updates or creates resources in the TADDM database.

Use the options similar to the update and create actions. Existing resources will be updated, and non-existing will be created.

If you use the --force (-f) argument you will not be prompted to confirm updates, which enables using this facility from a script.

`delete` Deletes resources from the TADDM database.

Use this powerful option with extreme care. Resources to be deleted are identified using the --list (-L), --groups (-C), --guids (-G), or type (-T) options.

If do not apply the --force (-f) argument, you will be prompted to confirm deletions.

`-A|--attributes <attribute>[:<new_value>] {,<attribute>[:<new_value>]}* } | <file_name>`

Specifies attributes, or attribute:value pairs to be used to control which attributes are listed/exported, and attribute values to be updated, or inserted.

-b|--basicauth — Uses header based HTTP authentication rather than the http authhandler class. Used internally, and has no effect on the operation of the tool.

`-B|--banner` — Show the input options

`-c|--compact` — Prevents printing of attributes for which the value is null.

`-C|--groups` — A comma separated list of GroupType:GroupName pairs, or a reference to a file that contains a list of GroupType:GroupName pairs, which is used to identify the members of the group(s) so they can be processed in one interaction.

Valid GroupTypes are BusinessSystem, Application,Collection, and AccessCollection.

This option is handy if a workload has been moved to a different site, and the monitoring agents need to update TADDM to reflect the new configuration.

`-d|--debug` — Provides information helpful when debugging the tool.

`-D|--distinct` — Instructs the rest_in_ease tools to select ONLY resources of the specified type. Similar to the ONLY clause in MQL queries.
If multiple used resource types are specified, the filter will be applied to each one individually, thus allowing you to create a report that contains multiple resource types from single query. Naturally, all the resource types must support all the attributes used in the filter.

`-e|--err` — Allows you to write the error messages to a file.

`-f|--force` — Used to avoid being prompted for confirmation to apply updates. This option is only enforced for the update, create, modify and delete actions.

`-F|--filter` — Used to qualify the instances of the resource types you are selecting. The filter is used to build the where clause of a MQL query.

`-g|--guidOnly` — Forces the export action to export only the guids of the selected resources. Applies only to the export action.

`-G|--guids` — A comma separated list of guids, or a reference to a file that contains a list of guids, which is used to identify resources to process.

`-h|--help` — Provides online help information.

`-H|--hostname` — The hostname or of the TADDM storage server you will connect to.

`-i|--input <file name>`
Reference to a file from which the tools reads input arguments. Specify the file name according to the syntax of your operating system, and relative to the path from which the tool in invoked.

`-k|--combined` — If used, this flag will be used to combine resources from multiple options, thereby allowing you to produce a list of resources from for example a list, a couple of guids, and specific resources identified by type and filter.

`-l|--log <file name>`
Allows you to write log information to a file. The file name provided is relative to the directory from which the tool is invoked.

`-L|--list <resourceType>:{<attribute:value>{,<attribute:value>}*} | <file_name>`
Used to produce a list of resources based on a combination of attribute-value pairs. The may be read from a flat file, which can be populated using the export action.

`-m|--members <groups|resources}`
Used in conjunction with --groups to decide which resources to show. Specifies which type of group resources to include in the report. You can select members or groups. If this argument is not specified, the report will show all group members.

`-M|--mss <mss_name>|<mss_guid>`
Use this argument in conjunction with the --type argument to select resources of the specified type that are registered with a specific management software system.

`-n|--noIDs` Applies only to the export action. Strips attributes such as guid, lastModifiedTime, lastModifiedBy, and type from the export data so they can be used by the create, attr, or list actions without modification.

`-o|--out <file name>`
Allows you to direct the output of the tool directly to a file. This option is mainly used in conjunction with the export action. The file name is relative to the current directory.

`-O|--orderBy <orderBy>`
Sorts the output in ascending order of the selected attribute. This option is only used for the list and attr actions.

`-p|--password` <password>
Password for the user used to authenticate with the TADDM storage server.

`-P|--port <port>` The port number you want to use for the connection to the TADDM storage server.
Default value is 9430.

`-q|--quiet` Suppresses console messages.

`-Q|--depth <depth>`
This integer controls the level of details from related resources that are collected from the TADDM database for each resource. The higher the number the more details. Default value is 1.

Use this option with extreme care. Gathering a lot of information about related resources may seriously impact the performance, and without providing a lot of benefit.

`-R|--registerMss [<mss_guid]> | <mss_name>`
Specifies the MSS that will be registered as responsible for the changes you apply. If option is specified and the mss_name or mss_guid is left blank, the changes will be registered with an MSS named TADDM_Manual_Update. If the mss_guid or mss_name has a value of “” (empty string) no MSS registration will be performed.

`-s|--shortFormat` Forces the class names of to be printed in the short format.

`-S|--fetchsize` `<max resources>`
Determines the maximum number of records to process. Default value is 1.

You should modify this option with care. Processing too may records may impact performance.

This option does not apply if you use the –C|--groups option to list all descendants of a group object.

`-t|--trace` Provides detailed information about the http calls performed by the tool.

If used in combination with –d or --debug, this will also trace the data parsed from one routine to another.

`-T|--types` A comma separated list of CDM class names of the resources to select – or of the resource types to include in a compound query

`-u|--user <username>`
Name of the user used to authenticate with the TADDM storage server.