Rocket MultiValue Integration Server

User Guide

Version 1.2.1

August 2019
MVIS-121-UG-01

Notices

Edition

Publication date: August 2019
Book number: MVIS-121-UG-01
Product version: Version 1.2.1

Copyright

Trademarks

Examples

This information might contain examples of data and reports. 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.

License agreement

Note: This product may contain encryption technology. Many countries prohibit or restrict the use, import, or export of encryption technologies, and current use, import, and export regulations should be followed when exporting this product.

Corporate information

Rocket Software, Inc. develops enterprise infrastructure products in four key areas: storage, networks, and compliance; database servers and tools; business information and analytics; and application development, integration, and modernization.

Website: www.rocketsoftware.com

Rocket Global Headquarters
77 4th Avenue, Suite 100
Waltham, MA 02451-1468
USA

To contact Rocket Software by telephone for any reason, including obtaining pre-sales information and technical support, use one of the following telephone numbers.

Country	Toll-free telephone number
United States	1-855-577-4323
Australia	1-800-823-405
Belgium	0800-266-65
Canada	1-855-577-4323
China	400-120-9242
France	08-05-08-05-62
Germany	0800-180-0882
Italy	800-878-295
Japan	0800-170-5464
Netherlands	0-800-022-2961
New Zealand	0800-003210
South Africa	0-800-980-818
United Kingdom	0800-520-0439

Contacting Technical Support

The Rocket Community is the primary method of obtaining support. If you have current support and maintenance agreements with Rocket Software, you can access the Rocket Community and report a problem, download an update, or read answers to FAQs. To log in to the Rocket Community or to request a Rocket Community account, go to www.rocketsoftware.com/support.

In addition to using the Rocket Community to obtain support, you can use one of the telephone numbers that are listed above or send an email to support@rocketsoftware.com.

Contents

Chapter 1: MultiValue Integration Server overview

Rocket MultiValue Integration Server (MVIS) is a piece of middleware that sits between an application and the MultiValue database server and practices cloud-friendly connection pooling, monitoring, and administration services.

MVIS manages and monitors API connections to the data servers.

MVIS also features a web-based user interface (MVIS Web Admin UI) and REST API into its administration and monitoring functions. MVIS requires a Java 8 runtime and can be installed on your data server machine, or on any machine that has network access to the data server.

Use MVIS to easily convert and expose your business logic (subroutines) or data resources as RESTful endpoint using the MVIS console or by using the MVIS Admin's APIs.

MVIS can provide an additional layer of resiliency and failover through the support of clustering and graceful shutdown.

The MVIS Web Admin UI is integrated with the Swagger OpenAPI 2.0. You can access the OpenAPI 2.0 definition for a specific endpoint directly from a subroutine or data resource definition. In addition, MVIS's administrative REST endpoints are also available through the Swagger UI for you to use in your own applications. These administrative OpenAPI endpoint definitions are available by clicking the Swagger icon from the bottom left-hand corner of the MVIS Web Admin UI.

MVIS also provides with the Swagger Editor which allows you to generate client- and server-side code to assist with mocking.

Chapter 2: Using MVIS with Docker containers

Docker is a container platform provider. Containers allow an application or its services and configurations to be packaged as a container image. This containerized application can then be tested as a unit and deployed as an image instance to the OS.

Each container image contains one application. Container images can then be created at each build or release without waiting until it's deployed. Creating container images at each build or release allows for environments to be consistent between a development and production environment.

Containers are deployed from the OS level rather than from a hardware level. OS level deployment allows containers to be isolated from other containers and the host. Each container has its own file system and acts independently of other containers. Since containers are not tied to a specific infrastructure or file system, containers are portable across different cloud environments and operating systems.

Deploying applications from the hardware level relies on users installing the application on a host that uses the operating system package manager. All application executables, libraries, and configuration are stored together, which can cause users to use the incorrect components. This type of deployment also relies on VMs, which are non-portable.

Containers are also capable of scalability. Users can create a new container for short-term tasks.

By default, there is no orchestration framework in place when running MVIS with Docker. Installing an orchestration framework, such as Kubernetes, allows users to automate the deployment process by setting up parameters to ensure that images are created under certain circumstances.

While this is helpful in most situation, users do not need to have an orchestration framework to run MVIS with Docker; however, users will have to manually deploy or create images when necessary.

Note: If you are using Kubernetes for Redis deployments based on cloud-specific persistent volumes, there are examples for Azure, AWS, Google in the `K8s_deployment_examples` folder of your installation download.

MVIS and Docker examples

In the following example, the MVIS image is run as a container using the Docker `run` command, passes in a Java options environment variable that will use Azure Blob storage and Azure Service Bus, and logs to Application Insights:

```
$ sudo docker run -p 7171:7171 -p 7870:7870 -p 7871:7871 -e "JAVA_OPTS=-DTYPE=MVCM\
-DBUCKET_NAME=cm-config -DCONFIG_NAME=cm.ini\
-DAZURE_SB_SAS_KEY=<Azure_Service_Bus_Key>\
-DAZURE_SB_NAMESPACE=<Azure_Service_Bus_Namespace>\
-DAZURE_ACCOUNT=<Azure_Storage_Account>\-DAZURE_ACCOUNT_KEY=<Azure_Storage_Account_Key> -
-DAPPLICATION_INSIGHTS_IKEY=<Azure_Application_Insights_Key>" cm
```

Note: When opening Docker containers in Windows, this syntax can be used from the Windows command prompt, but it is not valid syntax for the PowerShell prompt.

In the following example, the MVIS Admin image is run as a container using the Docker `run` command, passes in a Java options environment variable that will use Azure Blob storage and Azure Service Bus, and logs to Application Insights:

```
$ sudo docker run -p 7077:7077 -e\
"JAVA_OPTS=-DTYPE=MVCM -DBUCKET_NAME=cm-config\
```

```
-DCONFIG_NAME=cm.ini -DAZURE_SB_SAS_KEY=<Azure_Service_Bus_Key>\
-DAZURE_SB_NAMESPACE=<Azure_Service_Bus_Namespace>\
-DAZURE_ACCOUNT=<Azure_Storage_Account>\
-DAZURE_ACCOUNT_KEY=<Azure_Storage_Account_Key>\
-DCLOUD_LOGGING=AZURE -DAPPLICATION_INSIGHTS_IKEY=<Azure_Application_Insights_Key>"
cmadmin
```

Note: When opening Docker containers in Windows, this syntax can be used from the Windows command prompt, but it is not valid syntax for the PowerShell prompt.

Chapter 3: Starting and stopping MVIS

After you've installed MVIS, perform the following tasks to start or stop MVIS:

1. Starting the MultiValue Integration Server service on Windows
   There are two ways to start the MultiValue Integration Server service: from the Windows **Services** menu and from the MVIS Web Admin UI.
2. Starting the MVIS Admin on Windows
   On Windows, you can start and stop the MVIS Admin from the Microsoft Windows **Services** menu.
3. Opening the MVIS Web Admin UI
   After you've started the MVIS Admin, open the MVIS Web Admin UI.
4. Starting MVIS from the MVIS Web Admin UI
   Start MVIS from the MVIS Web Admin UI.
5. Stopping and shutting down MVIS
   Perform the following steps to stop and shut down MVIS.

Starting the MultiValue Integration Server service on Windows

There are two ways to start the MultiValue Integration Server service: from the Windows **Services** menu and from the MVIS Web Admin UI.

Prerequisites

Your UniVerse or UniData database must be running prior to starting MVIS.

Procedure

From the Windows Services window, find and select **MultiValue Integration Server** and click **Start the service**.

Next topic: Starting the MVIS Admin on Windows
Parent topic: Starting and stopping MVIS

Starting the MVIS Admin on Windows

On Windows, you can start and stop the MVIS Admin from the Microsoft Windows **Services** menu.

1. From the Microsoft Windows **Start** menu, navigate to **Services** → **MultiValue Integration Server Admin** → **Start the Service**.
2. To stop the MVIS Admin, navigate to **Services** → **MultiValue Integration Server Admin** → **Stop the Service**.

Next topic: Opening the MVIS Web Admin UI
Previous topic: Starting the MultiValue Integration Server service on Windows
Parent topic: Starting and stopping MVIS

Opening the MVIS Web Admin UI

After you've started the MVIS Admin, open the MVIS Web Admin UI.

1. Open a browser window.
2. In the address bar, enter `http://host:7077`.

   host is the location where MVIS is installed. 7077 is the default port for the MVIS Web Admin UI, but this port can be changed.
3. In the **Username** and **Password** field, enter your username and password. `admin` is the default for both fields.
4. Click **Sign In**.

   The MVIS Web Admin UI is displayed.

Next topic: Starting MVIS from the MVIS Web Admin UI
Previous topic: Starting the MVIS Admin on Windows
Parent topic: Starting and stopping MVIS

Starting MVIS from the MVIS Web Admin UI

Start MVIS from the MVIS Web Admin UI.

From the MVIS Web Admin UI, on the MultiValue Integration Server Status section, enter the server where MVIS is running, and then click **Start**.

> **Note:** If you are running in MVIS in a container environment, the container should be used to start the MVIS containers.

Next topic: Stopping and shutting down MVIS
Previous topic: Opening the MVIS Web Admin UI
Parent topic: Starting and stopping MVIS

Stopping and shutting down MVIS

Perform the following steps to stop and shut down MVIS.

About this task

> **Note:** If you are running in a container environment, the container orchestrator can be used to stop the MVIS container.

Procedure

1. If you are running MVIS on UNIX, complete the following steps:
   a. Enter the following command to shutdown MVIS from the Command Prompt:
      ```
      $ ./stop-cm.sh
      ```
2. If you are running MVIS on Windows, complete the following steps:
   a. Stop the MVIS Windows Service, or enter the following command to shutdown MVIS from the Command Prompt:
      ```
      C:\u2\cm> stop-cm.bat
      ```

Previous topic: Starting MVIS from the MVIS Web Admin UI
Parent topic: Starting and stopping MVIS

Chapter 4: Configuring MVIS

Use the MVIS Web Admin UI or `cm.ini` file to monitor accounts and performance and perform administration tasks.

Using the MVIS Web Admin UI, restart, disable, and delete accounts, edit account details, enable SSL, enable different forms of logging, and make other edits to a MVIS configuration.

You can also use MVIS to perform these functions with your own client through the MVIS Admin.

To configure MVIS, perform the following tasks:

- Adding an account to MVIS
  Once you've installed MVIS, create an account definition.
- Adding U2 REST server definitions
  From the MVIS Web Admin UI, you can add U2 REST server definitions.
- Converting your WebDE configuration file to a cm.ini configuration file
  You can convert an existing WebDE 5.3 `JavaScheduler.ini` configuration file to a MVIS `cm.ini` configuration file using the WebDE Configuration Conversion Tool (`wdeConverterTool.jar`).
- Modifying connection defaults
  You can check and edit the default connection details in the MVIS Web Admin UI.
- Monitoring connections
  MVIS features a performance monitor that records statistics on the requests processed for each account and displays results in a table. These statistics provide data to help you manage request processing and determine whether to make adjustments.
- Viewing MVIS logs
  You can view log information in the MVIS Web Admin UI and in individual log files stored in the MVIS directory.

Adding an account to MVIS

Once you've installed MVIS, create an account definition.

1. To start the MVIS Web Admin UI, enter the following URL in your web browser:
   http://localhost:7077
2. In the **User name** and **Password** fields, enter `admin`.
   The user name and password are `admin` by default, but they might have been changed.
3. From the **Menu** pane, select **Administration** → **Configuration** → **Accounts**.
4. Click **Add Account**.
5. In the Account Configuration window, enter the account details.
   For more information, see Configuring account settings, on page 12.
6. Click the **Test Connection** button to verify that MVIS is able to connect to this account.
7. Click **Okay**.
8. Click **Save**.
9. Click **Save**.

Parent topic: Configuring MVIS

Configuring account settings

After you've created an account, you can add additional settings.

1. From the MVIS Web Admin UI, select **Configuration** → **Accounts**, and then click the **Edit** icon.
2. In the Account Configuration window, click one of the following tabs:
   - **Server Details**
   - **Optional Settings**
   - **Monitor Warnings**
   - **Environment Variables**
3. In the **Server Details** tab, complete the following fields:

Field	Action
Account Name	Specify a name for the account. This does not have to match the name of the account on the data server.
MV Server Name	Specify the host name or IP address of the data server where the account resides.
MV Server Port	Specify the unirpc service port for the data server where the account resides by entering a value between 1 and 65535. The default value is 31438.
Protocol	Specify one of the following protocols that this account should use when being accessed from a client: ▪ **UNIOBJECTS**: Select this option to open the account pool to UOJ and UO.NET clients. This option is selected by default. ▪ **REST**: Select this option to open the account pool to REST clients.
Account Path	Specify the path to the MV account on the data server.
Connection String	Specify the entry from the `unirpcservices` file on the data server where the specified account is located. The default is `uvcs` for UniVerse and `udcs` for UniData.
User ID	Specify the operating system-level user ID of the user that will be used to connect to the data server for pooled connections to the account.
User Password	Specify the password to the operating system-level user ID that will be used to connect to the data server for pooled connections to the account. The password will be encrypted when saved.
SSL connection(s) to MV Server	Select this check box to activate SSL between MVIS and the data server hosting the account. Note, the database must be configured to allow SSL by the connection string service defined in the unirpcservice.
SSL hostname validation	When set to enabled, this specifies that the SSL connections, from MVIS to the data server, for the account require the hostname on the data server's SSL certificate match the server name for the account. By default, this is set to disabled.

Field	Action
Execution Timeout (seconds)	Specifies the number of seconds that the connection to the MV server will wait for a response before timing out and throwing an error.
MV Server Keep Alive	Select this check box to specify that a server-level keep alive heartbeat should be sent at a recurring interval between MVIS and the connections to the data server for the account. This option enables MVIS to keep its persistent connections alive, even when routers or cloud-based networks actively terminate idle connections. This option is enabled by default.
MV Server Keep Alive Interval (milliseconds)	Specify the interval, in milliseconds, to send a server-level keep alive heartbeat between MVIS and the connections to the data server for the account. The default is 60000 milliseconds.
Encoding	Specify the encoding for the connections to this account.
Minimum Pool Size	Specify the minimum number of connections in the connection pool for the account.
Maximum Pool Size	Specify the maximum number of connections maintained in the connection pool for the account.
Test Connection	Click this button to test your account connection settings.

4. In the **Optional Settings** tab, complete the following fields:

Field	Action
Service Definition Name	If you want to use an existing, shared REST server definition, type the name of the shared service that exists on another account.
Deactivate account	Select this check box to disable the account.
Enable server-side logging for this account	Select this check box to turn on COMO logs at the data server for the connections to the account.
Enable TCP Keep Alive between UO Clients and MVIS	Select this check box to turn on the **TCP keepalive socket** option between MVIS and its clients.
Enable development mode for this account	Select this check box to enable developer mode for this account. This allows all requests to get the latest version of compiled code rather than using a cached version of the object code. Development mode causes database processes to restart after each method call, while allowing newly compiled code to be picked up. This mode carries a performance overhead, as the associated database processes are ended and recreated after each method call. **Note:** Development mode is not recommended for production use.

5. In the **Monitor Warnings** tab, complete the following fields:

Field	Action
Average Response Time (Lifetime)	Specifies an average response time for requests. If this response time is exceed, the monitor warning level is triggered.
Average Response Time (Interval)	Specifies the average response time for requests to the account. If this response time is exceed during a monitor refresh interval, the monitor warning level is triggered.
Requests Waiting	Specifies the number of requests that are waiting for service. If this response time is exceed, the monitor warning level is triggered.
Slow Request	Specifies the amount of time that a process can run before the monitor warning level is triggered.

6. In the **Environment Variables** tab, perform the following steps to add a variable:
   a. Click **Add Variable**.
   b. In the **Variable Name** field, enter a name, and click **OK**.
   c. In the **Variable Value** field, enter a value for your variable, and then click **OK**.
   d. Click **OK**.
7. Click **OK**.

Adding U2 REST server definitions

From the MVIS Web Admin UI, you can add U2 REST server definitions.

There are two types of server definitions that you can create:

- Data resources
- Subroutines

When you create a REST endpoint that calls a subroutine, you can optionally specify a dynamic array structure for mapping to and from JSON for any of the subroutine parameters.

To add a U2 REST server definition, complete the following tasks:

- Creating REST data resources with MVIS
  Create a REST data resource if you want to create, read, update, or delete a file.
- Calling data resources with MVIS
  After you've created a data resource, use your browser to call the data resource.
- Creating REST subroutine definitions with MVIS
  You can create REST endpoints that invoke a BASIC subroutine on the data server.
- Calling BASIC subroutines with MVIS
- Sharing REST server definitions
  REST server definitions can be shared across multiple accounts. Complete these steps to share an existing REST server definition for an account to another account.
- Importing a U2 REST server
  You can import an existing U2 REST server into MVIS. When importing an existing U2 REST server, all associated Data Resources, Subroutines, Dynamic Arrays and security configurations are exposed to MVIS.
- Invoking the health check endpoint in U2REST
  MVIS provides a health check REST endpoint, which can be used to check if the MVIS REST server is running. Users can quickly check that the server is available.

Parent topic: Configuring MVIS

Creating REST data resources with MVIS

Create a REST data resource if you want to create, read, update, or delete a file.

1. From the MVIS Web Admin UI, select **Administration** → **REST Server** → **Accounts** → **Data Resources**.
2. Click **Add Data Resource**.
3. From the Data Resource window, enter the name of a U2 file in the **Data Resource** field.
4. In the **Data Resource Name** field, enter a name for the data resource.
5. Select the fields that you want to include in the data resource.
6. A foreign key is a field that is used in the TRANS virtual field to bring data from another file into the host. If you want to link your data resources from related files through the use of a foreign key, select the check box in the **F/K** column that is associated with the dictionary item you selected in the previous step.

   The linked file is displayed in the **Joined Files** tab.
7. To change the name of your linked file, from the **Joined Files** tab, enter a new name in the **Foreign Key** field.
8. For additional information on the sub-resources, click the **Data Subresources** tab.
9. Click **OK**.

Tip: To access the OpenAPI 2.0 definition for a specific endpoint, click the Swagger icon that is displayed in-line with the data resource definition. In addition, you can also use the MVIS administrative REST endpoints to create REST data resources. These administrative endpoints can be accessed by clicking the Swagger icon from the bottom left-hand corner of the MVIS Web Admin UI.

The Swagger Editor is also available for generating client- and server-side code to assist with mocking. To open the Swagger Editor, click the Swagger icon that is displayed in-line with the account name in the REST Server > Accounts section.

Parent topic: Adding U2 REST server definitions

Calling data resources with MVIS

After you've created a data resource, use your browser to call the data resource.

1. Open a browser window.
2. Enter the following URL:

   `http:\\`*`MVIS_location`*`:`*`port`*`\account\`*`data_resource_name`*

   where:
   - *MVIS_location* is the location of your MVIS.
   - *port* is the port number located in the **REST Server Status** section.
   - *account* is the account that is associated with the data resource.
   - *data_resource_name* is the name of the data resource.
3. Press Enter.

The data is returned in a JSON format in your browser.

Note: When calling data resource endpoints with select criteria, you must use spaces around the comparison operators in the select clause. For example, to select records with the last name of Smith, use the following format: http://server/account/file?select=lastName = “Smith”

Parent topic: Adding U2 REST server definitions

Creating REST subroutine definitions with MVIS

You can create REST endpoints that invoke a BASIC subroutine on the data server.

1. From the MVIS Web Admin UI, select **Administration** → **REST Server** → **Accounts** → **Subroutine Resources** → **Subroutines**, and then click **Add Subroutine**.
2. In the **Subroutine Details** tab from the Subroutine Service window, complete the following fields:

Field	Action
Subroutine Name	Enter the subroutine name.
Add Paramter	Click the **Add Parameter** button to add the required number of parameters for your subroutine.

3. Click the **Add Parameter** button to add the required number of parameters for your subroutine. Complete the following fields for each parameter:

Field	Action
Name	Specify a parameter name.
Parameter Type	Specify the type of parameter, whether **input**, **output** or **in/out**.
Data Type	Specify the data type for the parameter, whether **string**, **json** or **dynamic array**.
Dynamic Array Name	(Optional) Specify the name of a dynamic array associated with the input parameter.

Tip: Parameter positions in the list can be adjusted by dragging parameters up or down in the parameters list.

4. To use MVIS to optionally store BASIC subroutine source code in the subroutine definition file, which allows the MVIS Web Admin UI and Admin REST API to upload, compile, and catalog the subroutine source code to the specified account, in the **Optional Code** tab, complete the following fields:

Field	Action
Source Directory	Specifies the name of the directory where the source code is uploaded.

Field	Action
Create source directory if it doesn't exist	Specifies to create the source directory if a source directory does not exist.
Allow to compile and catalog subroutine code	Specifies whether to upload, compile, and catalog the source code from the **Source code** tab on a save action.
Compile Options	Specifies any optional compile flags that should be sent to the compile operation.
Catalog Options	Specifies any optional catalog flags that should be sent to the catalog operation.
Source code	Specifies the source code for the subroutine endpoint definition

5. Click **OK**.
6. To add a dynamic array, in the **Dynamic Arrays** section, click **Add Dynamic Array**.
7. In the Dynamic Array window, enter the name of the dynamic array in the **Dynamic Array Name** field, and then click **OK**.
8. Click **Add Field** and complete the following fields:

Field	Action
Name	Enter the name of the field in the dynamic array.
Location	Enter the location of the field in the dynamic array.
Type	Enter the type of field. Valid types are: ▪ s - Single-valued ▪ mv - Multivalued ▪ ms - Multi-subvalued
Association	Associations are tied to specific fields. If you've selected a field that contains an association, the association will be displayed in the **Association** column.
Sub Association	Sub-level of associations. Sub-associations will display in the **Sub Association** column.

9. To import a file dictionary as a new dynamic array definition, click the **Import** tab
10. From the **Import** tab, enter the name of a U2 file in the **U2 file name** field. Click **OK**.
11. Select the check box next to the dynamic array that you want to import, and click **Import**.
    The dynamic array is displayed in the **Dynamic Array Details** tab.
12. Click **OK**.

Tip: To access the OpenAPI 2.0 definition for a specific endpoint, click the Swagger icon that is displayed in-line with the subroutine resource definition. In addition, you can also use the MVIS administrative REST endpoints to create REST subroutine resources. These administrative endpoints can be accessed by clicking the Swagger icon from the bottom left-hand corner of the MVIS Web Admin UI.

The Swagger Editor is also available for generating client- and server-side code to assist with mocking. To open the Swagger Editor, click the Swagger icon that is displayed in-line with the account name in the REST Server > Accounts section.

Parent topic: Adding U2 REST server definitions

Calling BASIC subroutines with MVIS

Prerequisites

- Have a BASIC subroutine that is globally or locally cataloged and compiled in your U2 account. If a subroutine does not exist on the server, use the Optional tab in the Subroutine Editor.

Procedure

1. From the MVIS Web Admin UI, select **Administration** → **Configuration** → **Accounts**, and then click **Add Account**.
2. Create an account with the **Protocol** field set to **REST**.
3. Click **OK**.
4. Create a subroutine.
5. Verify your REST port by selecting **Administration** → **Configuration** → **Ports**. By default, the REST port is set to 7171.
6. Start MVIS.
7. In your browser, enter the URL, for example:

   ```
   http://localhost:7171/account_name/
   ```

   where *account_name* is the name of the account definitions created in Step 2.

   A list of available subroutines and data resources is displayed.
8. From the List Resource page, click the name of the subroutine to see the input fields, and then click **Call Service**.

   If the subroutine call was successful, the results will display. If it failed, an exception or error will be displayed.

Parent topic: Adding U2 REST server definitions

Sharing REST server definitions

REST server definitions can be shared across multiple accounts. Complete these steps to share an existing REST server definition for an account to another account.

These instructions assume that you have multiple accounts and an existing REST server definition on one of the accounts.

1. From the MVIS Web Admin UI, select **Administration** → **Configuration** → **Accounts**, and then click the **Edit account** button for the account that contains the REST server definition you want to share.
2. In the Account configuration dialog box, select the **Optional Settings** tab.
3. Type a name for the REST server definition to share in the **Service Definition Name** text box and click **OK**. This is the name you will use when sharing the REST server definition to anotherd account.
4. Click the **Edit account** button for the account to which you want to share the REST server definition.
5. In the Account configuration dialog box, select the **Optional Settings** tab.
6. Type the name of the shared REST server definition in the **Service Definition Name** text box and click **OK**. This is the same name you used when sharing the REST server definition from the first account.
7. Confirm that the REST server definition has been shared. From the Administration menu, select **REST Server** → **Accounts** and confirm that any Data Resources and Subroutine Resources defined on the sharing account now also exist on the shared to account.

Parent topic: Adding U2 REST server definitions

Importing a U2 REST server

You can import an existing U2 REST server into MVIS. When importing an existing U2 REST server, all associated Data Resources, Subroutines, Dynamic Arrays and security configurations are exposed to MVIS.

To import an existing U2 REST server, you must first export the REST server from the U2 RESTful Web Services Toolkit.

Note:

- U2 REST servers imported from the U2 RESTful Web Services Toolkit will have service definition files that are named after the U2 REST Resource Folder(s) hosted by the U2 REST server. In MVIS, service definition files are named after the cm.ini entry that a service is representing. As such, you should ensure that your cm.ini has an entry that corresponds to each U2 Resource Folder that you are importing from the U2 REST server. The one exception to this is if the cm.ini entry is using a Shared Definition. In such cases, the Shared Definition name should be represented by a U2 Resource Folder being imported.
- Only one U2 REST Server cay be imported into MVIS.

---

1. In the U2 RESTful Web Services Toolkit, right-click the REST server you want to export and select **Export**.

   The Export REST Server dialog box displays.
2. Select the REST server you want to export from the **Available REST Servers** check-boxes and then click **Finish**.
3. Open the MVIS Web Admin UI and select **REST Server** from the **MENU**.
4. Click **Browse...** from the Export/Import section and select the `<RESTServerName>.zip` file where *<RESTServerName>* is the name of the U2 REST Server that was exported from the U2 RESTful Web Services Toolkit.
5. Check the **Deployment from Web DE or RESTful Services Toolkit** check box, and specify the name of the rest server that you are importing from the U2 RESTful Web Services Toolkit.

Note: When importing from the <*RESTServerName*>.zip file, by default MVIS Admin avoids overwriting any existing service definition files. To override this behavior, click the **Force overwrite existing files** check box.

6. Click **Import**.

   The REST server is imported into MVIS and all associated Data Resources, Subroutines and Dynamic Arrays display in a new account on the REST Server page.

Note: The security configuration files (files with extensions .realm and .acl) are automatically re-encrypted upon import. This means that they will be decrypted using the U2 RESTful service algorithm and then re-encrypted using the MVIS encryption algorithm. The files names are also changed from *server_name* to *cm.rest* and are moved to the MVIS REST config location.

Parent topic: Adding U2 REST server definitions

Invoking the health check endpoint in U2REST

MVIS provides a health check REST endpoint, which can be used to check if the MVIS REST server is running. Users can quickly check that the server is available.

However, in most cases, an automated service will run these health checks at a preconfigured interval and take an automated action if a problem is detected. For example, if a server is acting as part of a cluster and it fails to respond, the server will be taken out of the cluster and automatically replaced by a new instance.

The MVIS REST server provides a health check endpoint at the following location:

```
http://host:port/healthcheck
```

This endpoint is automatically started when MVIS is configured to serve a REST API, so no configuration is required to enable the health check.

The endpoint can be invoked by a using a GET request. It will return an HTTP Status Code of `200 OK` and a payload of `OK`.

The following is an example of a health check request and response:

```
GET http://host:port/healthcheck

HTTP/1.1 200 OK
Content-Length: 2

OK
```

Parent topic: Adding U2 REST server definitions

Converting your WebDE configuration file to a cm.ini configuration file

You can convert an existing WebDE 5.3 `JavaScheduler.ini` configuration file to a MVIS `cm.ini` configuration file using the WebDE Configuration Conversion Tool (`wdeConverterTool.jar`).

The WebDE Configuration Conversion Tool is included in your MVIS installation package. It searches your WebDe installation directories for the `JavaScheduler.ini` file and then converts it to a MVIS `cm.ini` configuration file.

1. Locate the `JavaScheduler.ini` configuration file in your WebDE installation directory and make a note of the location.
2. Run the following command from the command prompt:

   ```
   java -DFS_DIR=JavaScheduler_Path -jar wdeConverterTool.jar
   ```

   where *JavaScheduler_Path* is the directory where the `JavaScheduler.ini` file resides.

   The program converts the `JavaScheduler.ini` file to a MVIS `cm.ini` configuration file.

   Note: If you don't know the location of the `JavaScheduler.ini` file on your system, you can run the following command instead:

   ```
   java -jar wdeConverterTool.jar
   ```

   This version of the command will search for the `JavaScheduler.ini` file in the following locations:

   - the current working folder
   - the path set in the U2WDE environment variable
   - the `C:\Windows` folder on Windows and the `/etc` folder on Unix

   Upon completion of the program, the new `cm.ini` file is saved in the same directory as the original `JavaScheduler.ini` file.

   Be aware that in many cases, names used for properties in the `JavaScheduler.ini` file have been changed in the `cm.ini` file. The table below lists the property names that have changed and indicates the new property name used in the `cm.ini` file:

   Table 1: JavaScheduler.in & cm.ini Property Mapping

JavaScheduler.ini	cm.ini
General Section	
IdleRemoveExecInterval	idleConnectionCheckInterval
MaxRestartWait	maxPoolRestartWaitTime
PoolingDebug	connectionTracingEnabled
SchedulerPort	connectionManagerPort
socketBackLog	uoMaxServerRequestQueue
usingTrustStore	trustStoreEnabled
sslKeyStore	sslKeyStorePath
sslTrustStore	sslTrustStorePath
ThreadSocketTimeOut	uoClientConnectionTimeout
AvgRespTime	avgRespTimeThresholds
AvgRespTimeIteration	respTimeThresholds
AcctRequestsQueued	requestsQueuedThresholds
usingssl	uoSSLEnabled
server	mvServer
Account Section	

JavaScheduler.ini	cm.ini
server	mvServer
MinimumPoolSize	minPoolSize
MaximumPoolSize	maxPoolSize
SlowProcessTime	slowRequestThreshold
AvgRespTime	avgRespTimeThresholds
AvgRespTimeIteration	respTimeThresholds
AcctRequestsQueued	requestsQueuedThresholds
unirpcPort	mvServerPort
uniRpcTimeout	mvServerConnectionTimeout
como	serverSideLoggingEnabled
devmode	devModeEnabled
tcpKeepAlive	uoClientTCPKeepAliveEnabled
usingssl	uoSSLEnabled

Parent topic: Configuring MVIS

Modifying connection defaults

You can check and edit the default connection details in the MVIS Web Admin UI.

1. From the MVIS Web Admin UI, select **Administration** → **Configuration**.
2. From the **MultiValue Integration Server Configuration** section, select one of the following sections:
   - **Ports**
   - **Control**
   - **Security**
   - **Logging**
3. In the **Ports** tab, complete the following fields:

Field	Action
MultiValue Integration Server Port	Specify the port number to access MVIS. The default port is 7870.
HTTP Port	Specify the port number associated with REST. The default port is 7171.
Monitor Port	Specify the port number for the monitor. The default port is 7871.

4. In the **Control** tab, complete the following steps:

Field	Action
Idle Connection Timeout (milliseconds)	Set the number of milliseconds that a connection to a backend data server can remain idle before it is flagged for removal. The default value is 60000.
Idle Connection Check Interval (milliseconds)	Set the number of milliseconds to wait before checking for and removing backend data server connections whose idle time has exceeded the value in **Idle Connection Timeout**. The default value is 60000.

Field	Action
Open Session Timeout (milliseconds)	Set the maximum amount of time to wait for a response from a data server connection before timing out. The default value is 1000.
Connection Tracing	This option logs detailed connection activity from MVIS to the data servers. This option is disabled by default.
Graceful Shutdown	This option sets whether the http server should be given time to gracefully shutdown before stopping MVIS. This option is disabled by default.
Graceful Shutdown Timeout (milliseconds)	Specify the timeout in milliseconds to use for the graceful shutdown of the http server when **Graceful Shutdown** is enabled. The default value is 300000.
REST Maximum Request Body Size (bytes).	Set the maximum size in bytes for a request payload for rest endpoints. The default is 200000 (200K bytes).
Rest Resources Cache Size (items)	Set the number of services that will be cached by the running MVIS after they are loaded from disk or other configuration storage. The default is 50.
HTTP Idle Timeout (milliseconds)	Specify the number milliseconds before an idle http connection from a REST client times out. The default is 30000 (30 seconds).
Max Server Request Queue (UO)	Specify the number of requests that can be queued on the listening socket when accepting connections from UO clients. The default value is 50 requests.

5. In the **Security** tab, complete the following fields:

Field	Action
Using SSL	Enable this option to use SSL on the MVIS's client-facing unirpc port. This option is disabled by default.
SSL HTTP(s)	Enable this option to use SSL on the MVIS's client-facing REST port. This option is disabled by default.
SSL Key Store Path	Enter the location of the SSL certificate for MVIS to protect client-facing connections.
SSL Key Store Password	Enter the password for the SSL keystore.
SSL Key Manager Password	Enter the password for the REST Jetty server's KeyManagerFactory. If not specified, the SSL keystore password is used.
Trust Store Enabled	Enable this option to use the truststore specified in **SSL Trust Store Path** field when validating secure connections to data servers. If this option is not enabled, a default store is used. By default, this option is disabled.
SSL Trust Store Path	Enter the path of the truststore that MVIS will use to validate secure connections to the data servers.

Field	Action
SSL Trust Store Password	Enter the truststore password.
Excluded Cipher Suites	Enter a space-delimited list of cipher suites to exclude when negotiating SSL.
Excluded Protocols	Enter a space-delimited list of protocols to exclude when negotiating SSL
HTTP Authentication	Specify the type of HTTP authentication to use when securing connections to REST services hosted from MVIS. The following options are available: ▪ http-basic ▪ http-digest ▪ none

6. In the **Logging** tab, complete the following fields:

Field	Action
Log Path	Specify the path to the `cm.log` file.
Log Levels	Select one of the following logging levels: ▪ `OFF` ▪ `TRACE` ▪ `DEBUG` ▪ `INFO` ▪ `WARNING` ▪ `ERROR` By default, Info is selected.

7. Click **Save**.

   Many of these details are created either by default or by user designation during the installation process, and they are stored in the `cm.ini` configuration file. When you make changes in the MVIS Web Admin UI, the changes are reflected in the `cm.ini` file, which is by default in the `cm` folder.

Parent topic: Configuring MVIS

Monitoring connections

MVIS features a performance monitor that records statistics on the requests processed for each account and displays results in a table. These statistics provide data to help you manage request processing and determine whether to make adjustments.

1. From the MVIS Web Admin UI, select **Administration** → **Configuration**.
2. In the **MultiValue Integration Server Status** section, in the **Host** field, enter the hostname of the MVIS that you want to monitor.
3. Click **Monitoring**.
4. In the **Monitor Control** area, click **Start**.

Your active account processes are displayed in a table in the lower portion of the MVIS Web Admin UI, where you can see requests, intervals, response times, and more. Slow processes are marked with yellow or red highlighting, depending on the severity of the slowdown. You can customize the triggers for fields being marked yellow and red; see Defining the yellow and red triggers of the monitor from the MVIS Web Admin UI, on page 25 or Defining the yellow and red triggers of the monitor from the cm.ini file, on page 25.

Additionally, you can view a BASIC call stack trace for each account shown in your monitor. Click the ellipses button to the right of an account name to view a stack trace for that account.

5. If performance statistic logging was not automatically started by the `startPerfStatsLoggingEnabled` property in `cm.ini` file, you can turn on performance statistics logging by completing the following steps:
   a. Expand the **Administration** section in the **Menu**.
   b. Click **Performance Statistics**.
   c. Change the default **Update Interval** by clicking the value next to the label, or accept the default.
   d. Check **Comma-Delimit Log** if you want the log to be comma-delimited, or leave the box unchecked if you prefer that the log is written in table format.
   e. Check **Log License Distribution** if you want to log only the distribution of the number of requests processed by each database session or license, or leave the box unchecked if you want all of the performance statistics to be logged.
   f. Click the **Start** button to begin logging performance statistics.

   Note: You cannot make changes to the **Update Interval**, **Comma-Delimit Log**, or **Log License Distribution** settings while the Performance Statistics Logger is running because the controls are disabled. Click the **Stop** button to make changes if the logger is already running, and then restart the logger.

Logs are written to the file `perfstats.log`, which is stored by default in the `cm` directory.

For information on logging while running in the cloud, see Using MVIS in the cloud, on page 30.

Parent topic: Configuring MVIS

Defining the yellow and red triggers of the monitor from the MVIS Web Admin UI

You can define the triggers for MVIS monitor from the MVIS Web Admin UI.

1. In the MVIS Web Admin UI, expand **Administration** in the **Menu**.
2. Select **Configuration**.
3. In the **Accounts** area, select the account for which you want to define triggers, and to the right of the account, click the **Edit accounts** button.
4. Click **Monitor warnings**.
5. Edit the trigger details for **Average Response Time (Lifetime)**, **Average Response Time (Interval)**, **Requests Waiting**, and **Slow Request** as desired.

Defining the yellow and red triggers of the monitor from the cm.ini file

The triggers for the MVIS monitor can be modified in the `cm.ini` file.

Open the `cm.ini` file in a text editor and add the following parameters to the appropriate account section:

Parameter	Description	Table field affected
`avgRespTimeThresholds=`*`n1 n2`*	Triggers monitor warnings when the overall average request response time exceeds these values. The average covers the period since the MVIS was last started. *n1* is the amount of time in milliseconds when the monitor displays yellow; *n2* is the amount of time in milliseconds when the monitor displays red.	Avg Response Time (Lifetime)
`requestsQueuedThresholds=`*`n1 n2`*	Triggers monitor warnings when the number of requests waiting to be served exceeds these values. *n1* is the amount of time in milliseconds when the monitor displays yellow; *n2* is the amount of time in milliseconds when the monitor displays red.	Requests Waiting
`respTimeThresholds=`*`n1 n2`*	Triggers monitor warnings when the average request response time exceeds these values. The average covers a single monitor refresh interval. *n1* is the amount of time in milliseconds when the monitor displays yellow; *n2* is the amount of time in milliseconds when the monitor displays red.	Avg Response Time (Interval)
`slowRequestThreshold=`*`n1`*	Triggers monitor warnings when any requests exceeds this value. *n1* is the amount of time in milliseconds when the monitor displays red.	Slow Request

Viewing MVIS logs

You can view log information in the MVIS Web Admin UI and in individual log files stored in the MVIS directory.

Use the MVIS Web Admin UI to view logs. The log information can also be sorted using the **Beginning**, **Next**, **End**, and **Errors** buttons.

You can also view log information in log files, which are stored in the `logs` folder in the `cm` directory by default. You can alter the default log location by changing the logpath property in the `cm.ini` file.

MVIS automatically creates some logs during certain processes. These log files generally are stored in the `logs` directory in the `cm` folder.

If you are running MVIS in a cloud environment, your logs can be stored in the cloud when MVIS is used in cloud-mode. For more information, see Using MVIS in the cloud, on page 30.

Select one of the following methods to configure logging:

- Configuring performance statistics with the cm.ini file
  Configure the `cm.ini` file to automatically log performance statistics prior to starting MVIS.
- Configuring performance statistics with the MVIS Web Admin UI
  Use the MVIS Web Admin UI to start logging performance statistics for MVIS.
- Configuring logging to the console on Linux/UNIX
  Setup logging to the console (standard output) when you are troubleshooting MVIS or the MVIS Admin. This can also be useful when running MVIS or MVIS Admin containers from a container orchestrator, such as Kubernetes, where the orchestrator takes data written to the console and provides an interface to inspect it while the container is running.
- Configuring logging to the console on Windows
  Setup logging to the console (standard output) when you are troubleshooting MVIS or the MVIS Admin. This can also be useful when running MVIS or MVIS Admin containers from a container orchestrator, such as Kubernetes, where the orchestrator takes data written to the console and provides an interface to inspect it while the container is running.

Parent topic: Configuring MVIS

Configuring performance statistics with the cm.ini file

Configure the `cm.ini` file to automatically log performance statistics prior to starting MVIS.

About this task

Use this method to configure logging if you have several instances of MVIS running in a cluster or you need to get statistics from MVIS to Application Insights or CloudWatch. If you use the MVIS Web Admin UI to configure logging, you will have to login to each MVIS Admin and switch on performance statistic logging.

Using the `cm.ini` file allows new cluster nodes to send the statistics to Application Insights or CloudWatch when you start MVIS without additional tuning.

This option can also be used if you are not running in a cluster or a cloud environment, but you want to start logging statistics after launching MVIS.

Procedure

1. Open the `cm.ini` file.
2. In the **[Default]** section of the `cm.ini` file, set the `startPerfStatsLoggingEnabled` to True.

   By default, the `startPerfStatsLoggingEnabled` property is set to False. When the `startPerfStatsLoggingEnabled` property is set to False, MVIS must be started and performance statistics logging must be turned on using the MVIS Web Admin UI.

   If the `startPerfStatsLoggingEnabled` is set to True, MVIS sends the statistics with the default values for the **Update Interval**, **Comma-Delimited Log**, and **Log License Distribution** fields. These values can only be changed in the MVIS Web Admin UI.

   Logs are saved to the `Perfstats.log` file, which is located in the `cm` directory.

   For information on logging to Application Insights, see Logging to Application Insights on Windows, on page 30 or Logging to Application Insights on Linux/UNIX, on page 31.

   For information on logging to CloudWatch, see Logging to Amazon CloudWatch on Windows, on page 37 or Logging to Amazon CloudWatch on Linux/UNIX, on page 37.

Parent topic: Viewing MVIS logs

Configuring performance statistics with the MVIS Web Admin UI

Use the MVIS Web Admin UI to start logging performance statistics for MVIS.

1. From the MVIS Web Admin UI, select **Administration** → **Performance Statistics**.
2. From the Performance Statistic Logging Control window, click **Start**.
3. To configure additional settings, complete the following fields:

Field	Action
Update Interval	Specifies the delay in seconds between the data provided in the log entries. By default, the update interval is set to 1 second.
Comma-Delimit Log	Select this check box to remove headers from the log and comma delimit the log entry. This option is ignored when performance statistics are sent to Azure Application Insights because values are sent as separate numeric metrics attached to a custom event. For information on logging to Azure Application Insights, see Logging to Application Insights on Windows, on page 30 or Logging to Application Insights on Linux/UNIX, on page 31. For information on logging to CloudWatch, see Logging to Amazon CloudWatch on Windows, on page 37 or Logging to Amazon CloudWatch on Linux/UNIX, on page 37.
Log License Distribution	Select this check box to log the distribution requests for U2 licenses.

Parent topic: Viewing MVIS logs

Viewing and setting MVIS log levels

You can view and adjust log levels for MVIS in the MVIS Web Admin UI.

Procedure

1. From the MVIS Web Admin UI, select **Administration** → **Configuration**.
2. Expand the **Log Level** section.
3. Review your log level setting and make any desired changes by selecting the appropriate radio button.
4. Click **Save**.

Configuring logging to the console on Linux/UNIX

Setup logging to the console (standard output) when you are troubleshooting MVIS or the MVIS Admin. This can also be useful when running MVIS or MVIS Admin containers from a container orchestrator, such as Kubernetes, where the orchestrator takes data written to the console and provides an interface to inspect it while the container is running.

1. To enable logging to the console for MVIS, complete the following steps:
   a. Navigate to the MVIS installation directory, and find the `start-cm.sh` script.
   b. Open `start-cm.sh` with a text editor, and then add the following lines to enable logging to the console:
      ```
      export JAVA_OPTS=-DTYPE=MVCM -DCONSOLE_LOGGING=1
      java $JAVA_OPTS -jar cm.jar
      ```
2. To enable logging to the console for the MVIS Admin, complete the following steps:
   a. Navigate to the MVIS installation directory and open the `application.properties` file with a text editor.
   b. In the **[Logging]** section, add the following property:
      ```
      logging.console.enabled=true
      ```

Parent topic: Viewing MVIS logs

Configuring logging to the console on Windows

Setup logging to the console (standard output) when you are troubleshooting MVIS or the MVIS Admin. This can also be useful when running MVIS or MVIS Admin containers from a container orchestrator, such as Kubernetes, where the orchestrator takes data written to the console and provides an interface to inspect it while the container is running.

1. To enable logging to the console for MVIS, complete the following steps:
   a. Navigate to the MVIS installation directory, and find the `start-cm.bat` script.
   b. Open `start-cm.bat` with a text editor, and then add the following lines to enable logging to the console:
      ```
      set JAVA_OPTS=-DTYPE=MVCM -DCONSOLE_LOGGING=1
      java %JAVA_OPTS% -jar "c:\u2\cm\cm.jar"
      ```
2. To enable logging to the console for the MVIS Admin, complete the following steps:
   a. Navigate to the MVIS installation directory and open the `application.properties` file with a text editor.
   b. In the **[Logging]** section, add the following property:
      ```
      logging.console.enabled=true
      ```

   Logs from MVIS are now sent to the console.

Parent topic: Viewing MVIS logs

Chapter 5: Using MVIS in the cloud

Cloud-based services can be used to build, deploy, and manage applications through a global network of data centers.

MVIS is available on the following cloud services:

- Azure overview
  Microsoft Azure is a series of cloud-based services that can be used to build, deploy, and manage applications through a global network of data centers.
- AWS overview
  Amazon Web Services (AWS) is a secure cloud services platform.

Azure overview

Microsoft Azure is a series of cloud-based services that can be used to build, deploy, and manage applications through a global network of data centers.

When working with Azure, the following definitions are used:

Application Insights

A unified console that simplifies building, deploying, and managing cloud resources.

Application Insights is also an analytic service that helps users understand the performance and usage of their live web applications.

Blob storage

An object storage service that can be used for cloud applications, content distribution, backups, archiving, and disaster recovery.

Service bus

Service bus allows MVIS clusters to communicate via messaging.

Shared Access Signatures (SAS)

SAS is the primary security mechanism for service bus messaging and guards the service bus based on authorization rules. These rules are configured on a namespace or messaging entity (relay, queue, or topic).

Parent topic: Using MVIS in the cloud

Logging to Application Insights on Windows

Configure MVIS to send log records to Application Insights.

1. To enable logging to Application Insights, navigate to the installation directory and open the `start-cm.bat` file.
2. Using a text editor, modify the `start-cm.bat` file to contain the following code:

```
java -DTYPE=MVCM^
  -DCLOUD_LOGGING=AZURE\^
  -DAPPLICATION_INSIGHTS_IKEY=<application_insights_instrumentation_key>^
  -jar cm.jar
```

   Your program will now send a copy of the log to Application Insights.

   Note, if a user enables performance statistics from the MVIS Web Admin UI with these Java options specified, performance statistics will also be sent to Application Insights.

Logging to Application Insights on Linux/UNIX

Configure MVIS to send log records to Application Insights.

1. To enable logging to Application Insights, navigate to the installation directory and open the `start-cm.sh` file.
2. Using a text editor, modify the `start-cm.sh` file to contain the following code:

```
$ java -DTYPE=MVCM\
 -DCLOUD_LOGGING=AZURE\
 -DAPPLICATION_INSIGHTS_IKEY=<application_insights_instrumentation_key>\
 -jar cm.jar
```

   Your program will now send a copy of the log to Application Insights.

   Note, if a user enables performance statistics from the MVIS Web Admin UI with these Java options specified, performance statistics will also be sent to Application Insights.

Sending performance statistics to Application Insights on Linux/UNIX

Application Insights is an application performance management (APM) service that can be used to monitor live web applications and detect any performance issues. Configure MVIS to send performance statistic entries to Application Insights.

1. To send performance statistics to Application Insights, navigate to the installation directory and open the `start-cm.sh` file.
2. Using a text editor, modify the `start-cm.sh` file to contain the following code:

```
$ java -DTYPE=MVCM\
  -DAPPLICATION_INSIGHTS_IKEY=<application_insights_instrumentation_key>\
  -jar cm.jar
```

3. Using a web browser, open the MVIS Web Admin UI.
4. From the MVIS Web Admin UI, select **Administration** → **Performance Statistics**, and then click **Start**.

   Your program will now send performance statistic entries to Application Insights.

Configuring MVIS and the MVIS Admin for Azure Blob storage

MVIS can store its configuration data in Azure Blob storage. This is especially useful when running a cluster of MVISs that should all share a common configuration.

To configure MVIS and the MVIS Admin to store data in Azure Blob storage, complete the following tasks:

- Configuring MVIS for Blob storage on Linux/UNIX

  MVIS can be configured to use Blob storage as its configuration store by specifying additional Java system properties when starting MVIS. The properties identify which Azure storage account to use, what container name to store the configuration files under, the name of the main configuration file, and the Azure storage account key to use.
- Configuring the MVIS Admin for Blob storage on Linux/UNIX

  Configure the MVIS Admin to use Blob storage to store MVIS configuration files. This includes the main MVIS configuration file and your U2 REST definitions.
- Configuring MVIS for Blob storage on Windows

MVIS can be configured to use Blob storage as its configuration store by specifying additional Java system properties when starting MVIS. The properties identify which Azure storage account to use, what container name to store the configuration files under, the name of the main configuration file, and the Azure storage account key to use.

- Configuring the MVIS Admin for Blob storage on Windows
  Configure the MVIS Admin to use Blob storage to store MVIS configuration files. This includes the main MVIS configuration file and your U2 REST definitions.

Configuring MVIS for Blob storage on Linux/UNIX

MVIS can be configured to use Blob storage as its configuration store by specifying additional Java system properties when starting MVIS. The properties identify which Azure storage account to use, what container name to store the configuration files under, the name of the main configuration file, and the Azure storage account key to use.

About this task

For more information on how to create an Azure storage account, see the Azure documentation at https://docs.microsoft.com/en-us/azure/.

Procedure

1. To configure MVIS to use Blob storage, navigate to the installation directory and open the `start-cm.sh` file.
2. Using a text editor, modify the `start-cm.sh` file to contain the following code:

```
$ java -DTYPE=MVCM\
  -DAZURE_ACCOUNT=<Azure_Storage_Account>\
  -DAZURE_ACCOUNT_KEY=<Azure_Storage_Account_Key>\
  -DAZURE_SB_NAMESPACE=<Azure_service_bus_namespace>\
  -DAZURE_SB_SAS_KEY=<Azure_service_bus_sas_key>\
  -DBUCKET_NAME=<container_name>\
  -DCONFIG_NAME=<main_config_file_name>\
  -jar cm.jar
```

Parent topic: Configuring MVIS and the MVIS Admin for Azure Blob storage

Configuring the MVIS Admin for Blob storage on Linux/UNIX

Configure the MVIS Admin to use Blob storage to store MVIS configuration files. This includes the main MVIS configuration file and your U2 REST definitions.

About this task

The properties identify which Azure storage account, container name to store the configuration files, name of the main configuration file, and which Azure storage account key to use.

For more information on how to create an Azure storage account, see the Azure documentation at https://docs.microsoft.com/en-us/azure/.

Procedure

1. To configure the MVIS Admin, navigate to the installation directory and open the `start-admin.sh` file.
2. Using a text editor, modify the `start-admin.sh` file to contain the following code:

a. From the Command Prompt, enter the following command:

```
$ java -DTYPE=MVCM\
  -DAZURE_ACCOUNT=<Azure_Storage_Account>\
  -DAZURE_ACCOUNT_KEY=<Azure_Storage_Account_Key>\
  -DAZURE_SB_NAMESPACE=<Azure_service_bus_namespace>\
  -DAZURE_SB_SAS_KEY=<Azure_service_bus_sas_key>\
  -DBUCKET_NAME=<container_name>\
  -DCONFIG_NAME=<main_config_file_name>\
  -jar cm-admin.jar
```

Parent topic: Configuring MVIS and the MVIS Admin for Azure Blob storage

Configuring MVIS for Blob storage on Windows

MVIS can be configured to use Blob storage as its configuration store by specifying additional Java system properties when starting MVIS. The properties identify which Azure storage account to use, what container name to store the configuration files under, the name of the main configuration file, and the Azure storage account key to use.

About this task

For more information on how to create an Azure storage account, see the Azure documentation at https://docs.microsoft.com/en-us/azure/.

Procedure

1. To configure MVIS to use Blob storage, navigate to the installation directory and open the `start-cm.bat` file.
2. Using a text editor, modify the `start-cm.bat` file to contain the following code:

```
java -DTYPE=MVCM^
  -DAZURE_ACCOUNT=<Azure_Storage_Account>^
  -DAZURE_ACCOUNT_KEY=<Azure_Storage_Account_Key>^
  -DAZURE_SB_NAMESPACE=<Azure_service_bus_namespace>^
  -DAZURE_SB_SAS_KEY=<Azure_service_bus_sas_key>^
  -DBUCKET_NAME=<container_name>^
  -DCONFIG_NAME=<main_config_file_name>^
  -jar cm.jar
```

Parent topic: Configuring MVIS and the MVIS Admin for Azure Blob storage

Configuring the MVIS Admin for Blob storage on Windows

Configure the MVIS Admin to use Blob storage to store MVIS configuration files. This includes the main MVIS configuration file and your U2 REST definitions.

About this task

The properties identify which Azure storage account, container name to store the configuration files, name of the main configuration file, and which Azure storage account key to use.

For information on how to create an Azure Service Bus namespace and a shared access signature, see the Azure documentation at https://docs.microsoft.com/en-us/azure/.

Procedure

1. To configure the MVIS Admin to use Blob storage, navigate to the installation directory and open the `cm-admin.xml` file.

2. Edit the arguments element in the `cm-admin.xml` file to look like the following:

```
<service>
  <id>cm-admin</id>
  <name>Connection Manager Admin</name>
  <description>This service runs CM-ADMIN system.</description>
  <executable>java</executable>
  <arguments>-DTYPE=MVCM -jar
     -DAZURE_ACCOUNT=<Azure_Storage_Account>
     -DAZURE_ACCOUNT_KEY=<Azure_Storage_Account_Key>
     -DAZURE_SB_NAMESPACE=<Azure_service_bus_namespace>
     -DAZURE_SB_SAS_KEY=<Azure_service_bus_sas_key>
     -DBUCKET_NAME=<container_name>
     -DCONFIG_NAME=<main_config_file_name>"%BASE%\
  cm-admin.jar"</arguments>
  <logmode>rotate</logmode>
  <logpath>%BASE%\logs</logpath>
```

Parent topic: Configuring MVIS and the MVIS Admin for Azure Blob storage

Configuring MVIS and the MVIS Admin for Azure Service Bus

MVIS can store configurations in cloud-based data stores, such as Azure Blob storage. In these scenarios, the MVIS Admin saves configuration changes directly to the cloud-based data stores. When a configuration change is saved, the MVIS Admin also publishes a message to a cloud-based message broker that indicates a configuration change occurred. MVIS waits for such messages and, if it receives such a notification, will reload its configuration.

A cluster of load-balanced MVISs will all receive the messages published from the MVIS Admin. This allows a clustered set of MVISs to act in unison to the changes published by the MVIS Admin.

The following configuration changes will take effect immediately in the running MVIS:

- Any newly added accounts will have their connection pools started.
- Any removed accounts will have their connection pools shutdown.
- Certain changes to an existing account will cause that account's pool to be restarted with the new parameters.

To configure MVIS and the MVIS Admin to use Azure Service Bus, complete the following tasks:

- Configuring MVIS for Azure Service Bus on Linux/UNIX
  MVIS can be configured to use Service Bus as its message broker by specifying additional Java system properties when starting MVIS. The properties identify which Service Bus namespace and Service Bus shared access signature key MVIS uses.
- Configuring the MVIS Admin for Azure Service Bus on Linux/UNIX
  The MVIS Admin can be configured to use Service Bus as its message broker. These properties identify which Service Bus namespace and shared access signature key the MVIS Admin uses.
- Configuring MVIS for Azure Service Bus on Windows
  MVIS can be configured to use Service Bus as its message broker by specifying additional Java system properties when starting MVIS. The properties identify which Service Bus namespace and Service Bus shared access signature key MVIS uses.
- Configuring the MVIS Admin for Azure Service Bus on Windows
  The MVIS Admin can be configured to use Service Bus as its message broker. These properties identify which Service Bus namespace and shared access signature key the MVIS Admin uses.

Configuring MVIS for Azure Service Bus on Linux/UNIX

MVIS can be configured to use Service Bus as its message broker by specifying additional Java system properties when starting MVIS. The properties identify which Service Bus namespace and Service Bus shared access signature key MVIS uses.

About this task

For more information on how to create a Service Bus namespace and shared access signature, see the Azure documentation at https://docs.microsoft.com/en-us/azure/.

Procedure

1. To configure Service Bus as the message broker for MVIS, navigate to the installation directory and open the `start-cm.sh` file.
2. Using a text editor, modify the `start-cm.sh` file to contain the following code:

```
$ java -DTYPE=MVCM\
  -DAZURE_SB_SAS_KEY=<Azure_Service_Bus_Shared_Access_Signature_Key>\
  -DAZURE_SB_NAMESPACE=<Azure_Service_Bus_Namespace>\
  -jar cm.jar
```

Parent topic: Configuring MVIS and the MVIS Admin for Azure Service Bus

Configuring the MVIS Admin for Azure Service Bus on Linux/UNIX

The MVIS Admin can be configured to use Service Bus as its message broker. These properties identify which Service Bus namespace and shared access signature key the MVIS Admin uses.

About this task

For more information on how to create a Service Bus namespace and shared access signature, see the Azure documentation at https://docs.microsoft.com/en-us/azure/.

Procedure

1. To configure the MVIS Admin, navigate to the installation directory and open the `start-admin.sh` file.
2. Using a text editor, modify the `start-admin.sh` file to contain the following code:

```
$ java -DTYPE=MVCM\
  -DAZURE_SB_NAMESPACE=<Azure_Service_Bus_Namespace>\
  -DAZURE_SB_SAS_KEY=<Azure_Service_Bus_Shared_Access_Signature_Key>\
  -jar cm-admin.jar
```

Parent topic: Configuring MVIS and the MVIS Admin for Azure Service Bus

Configuring MVIS for Azure Service Bus on Windows

MVIS can be configured to use Service Bus as its message broker by specifying additional Java system properties when starting MVIS. The properties identify which Service Bus namespace and Service Bus shared access signature key MVIS uses.

About this task

For more information on how to create a Service Bus namespace and shared access signature, see the Azure documentation at https://docs.microsoft.com/en-us/azure/.

Procedure

1. To configure Service Bus as the message broker for MVIS, navigate to the installation directory and open the `start-cm.bat` file.
2. Using a text editor, modify the `start-cm.bat` file to contain the following code:

```
java -DTYPE=MVCM^
  -DAZURE_SB_SAS_KEY=<Azure_Service_Bus_Shared_Access_Signature_Key>^
  -DAZURE_SB_NAMESPACE=<Azure_Service_Bus_Namespace>^
  -jar cm.jar
```

Parent topic: Configuring MVIS and the MVIS Admin for Azure Service Bus

Configuring the MVIS Admin for Azure Service Bus on Windows

The MVIS Admin can be configured to use Service Bus as its message broker. These properties identify which Service Bus namespace and shared access signature key the MVIS Admin uses.

About this task

For more information on how to create a Service Bus namespace and shared access signature, see the Azure documentation at https://docs.microsoft.com/en-us/azure/.

Procedure

1. To configure the MVIS Admin, navigate to the installation directory and open the `cm-admin.xml` file.
2. Edit the arguments element in the `cm-admin.xml` file to look like the following:

```
<service>
  <id>cm-admin</id>
  <name>Connection Manager Admin</name>
  <description>This service runs CM-ADMIN system.</description>
  <executable>java</executable>
  <arguments>-DTYPE=MVCM -jar
     -DAZURE_SB_NAMESPACE=<Azure_Service_Bus_Namespace>
     -DAZURE_SB_SAS_KEY=<Azure_Service_Bus_Shared_Access_Signature_Key>
   "%BASE%\
  cm-admin.jar"</arguments>
  <logmode>rotate</logmode>
  <logpath>%BASE%\logs</logpath>
```

3. Restart the MVIS Admin service for your changes to take effect.

Parent topic: Configuring MVIS and the MVIS Admin for Azure Service Bus

AWS overview

Amazon Web Services (AWS) is a secure cloud services platform.

When working with AWS, the following definitions are used:

Amazon CloudWatch

Amazon CloudWatch is a monitoring service that supports centralized logging against a cluster of MultiValue Integration Servers.

Simple Notification Service (SNS)

SNS is a service that provides event notifications that users subscribed to and allows MVIS clusters to communicate via messaging.

Simple Storage Service (S3)

An object storage service that is used to hold MVIS configuration files.

Parent topic: Using MVIS in the cloud

Logging to Amazon CloudWatch on Windows

Amazon CloudWatch is a monitoring service for AWS and your applications that run on AWS. Use Amazon CloudWatch to monitor your performance and react to changes in your AWS resources.

1. To enable logging to Amazon CloudWatch, navigate to the installation directory and open the `start-cm.bat` file.
2. Using a text editor, modify the `start-cm.bat` file to contain the following code:

```
java -DTYPE=MVCM^
  -DCLOUD_LOGGING=AWS^
  -DAWS_REGION=<aws_region>^
  -DAWS_ACCESS_KEY=<aws_access_key>^
  -DAWS_SECRET_KEY=<aws_secret_key>^
  -jar cm.jar
```

3. You can also add logging configuration options to distinguish between log sources:
   - **Log Group**: A log group is a group of log streams that share the same retention, monitoring, and access control settings. You can define log groups and specify which streams to put into each group. There is no limit on the number of log streams that can belong to one log group. If not specified, the default log group name will be `cm`.
   - **Log Stream**: A log stream is a sequence of log events that share the same source. Each separate source of logs into CloudWatch Logs makes up a separate log stream. If not specified, the default log stream name will be `cm-main-logs`.

     Specify these options by adding the code below:

```
java -DTYPE=MVCM^
  -DCLOUD_LOGGING=AWS^
  -DAWS_REGION=<aws_region>^
  -DAWS_ACCESS_KEY=<aws_access_key>^
  -DAWS_SECRET_KEY=<aws_secret_key>^
  -DAWS_LOG_GROUP=<aws_log_group>^
  -DAWS_LOG_STREAM=<aws_log_stream>^
  -jar cm.jar
```

Your program will now send a copy of the log to Amazon CloudWatch.

Logging to Amazon CloudWatch on Linux/UNIX

Amazon CloudWatch is a monitoring service for AWS and your applications that run on AWS. Use Amazon CloudWatch to monitor your performance and react to changes in your AWS resources.

1. To enable logging to Amazon CloudWatch, navigate to the installation directory and open the `start-cm.sh` file.
2. Using a text editor, modify the `start-cm.sh` file to contain the following code:

```
$ java -DTYPE=MVCM\
  -DCLOUD_LOGGING=AWS\
  -DAWS_REGION=<aws_region>\
  -DAWS_ACCESS_KEY=<aws_access_key>\
  -DAWS_SECRET_KEY=<aws_secret_key>\
```

```
-jar cm.jar
```

3. You can also add logging configuration options to distinguish between log sources:
   - **Log Group**: A log group is a group of log streams that share the same retention, monitoring, and access control settings. You can define log groups and specify which streams to put into each group. There is no limit on the number of log streams that can belong to one log group. If not specified, the default log group name will be `cm`.
   - **Log Stream**: A log stream is a sequence of log events that share the same source. Each separate source of logs into CloudWatch Logs makes up a separate log stream. If not specified, the default log stream name will be `cm-main-logs`.

     Specify these options by adding the code below:

```
$ java -DTYPE=MVCM\
  -DCLOUD_LOGGING=AWS\
  -DAWS_REGION=<aws_region>\
  -DAWS_ACCESS_KEY=<aws_access_key>\
  -DAWS_SECRET_KEY=<aws_secret_key>\
  -DAWS_LOG_GROUP=<aws_log_group>\
  -DAWS_LOG_STREAM=<aws_log_stream>\
  -jar cm.jar
```

Your program will now send a copy of the log to Amazon CloudWatch.

Sending performance statistics to AWS CloudWatch on Windows

CloudWatch is an application performance management (APM) service used to monitor live web applications and detect performance issues.

1. To configure MVIS to send performance statistics to AWS CloudWatch, navigate to the installation directory and open the `start-cm.bat` file.
2. Using a text editor, modify the `start-cm.bat` file to contain the following code:

```
$ java -DTYPE=MVCM^
  -DAWS_REGION=<aws_region>^
  -DAWS_ACCESS_KEY=<aws_access_key>^
  -DAWS_SECRET_KEY=<aws_secret_key>^
  -jar cm.jar
```

3. You can also add CloudWatch namespaces, which contain metrics. If not specified, the default namespace will be `CM_PERFORMANCE_STATISTICS`. For example:

```
$ java -DTYPE=MVCM^
   -DAWS_REGION=<aws_region>^
   -DAWS_ACCESS_KEY=<aws_access_key>^
   -DAWS_SECRET_KEY=<aws_secret_key>^
   -DAWS_CW_NAMESPACE=<aws_cw_namespace>^
   -jar cm.jar
```

4. Open the MVIS Web Admin UI.
5. From the **Administration** menu, select **Performance Statistics**.
6. Click the **Start** button to start logging performance statistics.

   The program will now send performance statistic entries to AWS CloudWatch.

Sending performance statistics to AWS CloudWatch on Linux / UNIX

CloudWatch is an application performance management (APM) service used to monitor live web applications and detect performance issues.

1. To configure MVIS to send performance statistics to AWS CloudWatch, navigate to the installation directory and open the `start-cm.sh` file.
2. Using a text editor, modify the `start-cm.sh` file to contain the following code:

```
$ java -DTYPE=MVCM\
   -DAWS_REGION=<aws_region>\
   -DAWS_ACCESS_KEY=<aws_access_key>\
   -DAWS_SECRET_KEY=<aws_secret_key>\
   -jar cm.jar
```

3. You can also add CloudWatch namespaces, which contain metrics. If not specified, the default namespace will be `CM_PERFORMANCE_STATISTICS`. For example:

```
$ java -DTYPE=MVCM\
   -DAWS_REGION=<aws_region>\
   -DAWS_ACCESS_KEY=<aws_access_key>\
   -DAWS_SECRET_KEY=<aws_secret_key>\
   -DAWS_CW_NAMESPACE=<aws_cw_namespace>\
   -jar cm.jar
```

4. Open the MVIS Web Admin UI.
5. From the **Administration** menu, select **Performance Statistics**.
6. Click the **Start** button to start logging performance statistics.

   The program will now send performance statistic entries to AWS CloudWatch.

Configuring MVIS and the MVIS Admin for Amazon S3 storage

MVIS can store its configuration data in Amazon S3 storage. This is especially useful when running a cluster of MultiValue Integration Servers that should all share a common configuration.

To configure MVIS and the MVIS Admin data in Amazon S3 storage, complete the following tasks:

- Configuring MVIS for Amazon S3 storage on Linux/UNIX

  MVIS can be configured to use Amazon S3 storage as its configuration store by specifying additional Java system properties when starting MVIS. These properties identify what AWS access key, secret key, and AWS region that MVIS will use.
- Configuring the MVIS Admin for Amazon S3 storage on Linux/UNIX

  The MVIS Admin can be configured to use Amazon S3 storage. These properties identify what AWS access key, secret key, and AWS region MVIS will use.
- Configuring MVIS for Amazon S3 storage on Windows

  MVIS can be configured to use Amazon S3 storage as its configuration store by specifying additional Java system properties when starting MVIS. These properties identify what AWS access key, secret key, and AWS region that MVIS will use.
- Configuring the MVIS Admin for Amazon S3 storage on Windows

  The MVIS Admin can be configured to use Amazon S3 storage. These properties identify what AWS access key, secret key, and AWS region MVIS will use.

Configuring MVIS for Amazon S3 storage on Linux/UNIX

MVIS can be configured to use Amazon S3 storage as its configuration store by specifying additional Java system properties when starting MVIS. These properties identify what AWS access key, secret key, and AWS region that MVIS will use.

About this task

For more information on Amazon S3, including how to set up AWS access keys, secret keys, and AWS regions, see the AWS documentation at https://aws.amazon.com/documentation/.

Procedure

1. To configure MVIS to use Amazon S3 storage, navigate to the installation directory and open the `start-cm.sh` file.
2. Using a text editor, modify the `start-cm.sh` file to contain the following code:

```
$ java -DTYPE=MVCM\
  -DAWS_ACCESS_KEY=<AWS_Access_Key>\
  -DAWS_SECRET_KEY=<AWS_Secret_Key>\
  -DAWS_REGION=<AWS_Region>\
  -DBUCKET_NAME=<bucket_name>\
  -DCONFIG_NAME=<main_config_file_name>\
  -jar cm.jar
```

Parent topic: Configuring MVIS and the MVIS Admin for Amazon S3 storage

Configuring the MVIS Admin for Amazon S3 storage on Linux/UNIX

The MVIS Admin can be configured to use Amazon S3 storage. These properties identify what AWS access key, secret key, and AWS region MVIS will use.

About this task

For more information on Amazon S3, including how to set up AWS access keys, secret keys, and AWS regions, see the AWS documentation at https://aws.amazon.com/documentation/.

Procedure

1. To configure the MVIS Admin, navigate to the installation directory and open the `start-admin.sh` file.
2. Using a text editor, modify the `start-admin.sh` file to contain the following code:

```
$ java -DTYPE=MVCM\
  -DAWS_ACCESS_KEY=<AWS_Access_Key>\
  -DAWS_SECRET_KEY=<AWS_Secret_Key>\
  -DAWS_REGION=<AWS_Region>\
  -DBUCKET_NAME=<bucket_name>\
  -DCONFIG_NAME=<main_config_file_name>\
  -jar cm-admin.jar
```

Parent topic: Configuring MVIS and the MVIS Admin for Amazon S3 storage

Configuring MVIS for Amazon S3 storage on Windows

MVIS can be configured to use Amazon S3 storage as its configuration store by specifying additional Java system properties when starting MVIS. These properties identify what AWS access key, secret key, and AWS region that MVIS will use.

About this task

For more information on Amazon S3, including how to set up AWS access keys, secret keys, and AWS regions, see the AWS documentation at https://aws.amazon.com/documentation/.

Procedure

1. To configure MVIS to use Amazon S3 storage, navigate to the installation directory and open the `start-cm.bat` file.
2. Using a text editor, modify the `start-cm.bat` file to contain the following code:

```
java -DTYPE=MVCM^
  -DAWS_ACCESS_KEY=<AWS_Access_Key>^
  -DAWS_SECRET_KEY=<AWS_Secret_Key>^
  -DAWS_REGION=<AWS_Region>^
  -DBUCKET_NAME=<bucket_name>^
  -DCONFIG_NAME=<main_config_file_name>^
  -jar cm.jar
```

Parent topic: Configuring MVIS and the MVIS Admin for Amazon S3 storage

Configuring the MVIS Admin for Amazon S3 storage on Windows

The MVIS Admin can be configured to use Amazon S3 storage. These properties identify what AWS access key, secret key, and AWS region MVIS will use.

About this task

For more information on Amazon S3, including how to set up AWS access keys, secret keys, and AWS regions, see the AWS documentation at https://aws.amazon.com/documentation/.

Procedure

1. To configure the MVIS Admin, navigate to the installation directory and open the `cm-admin.xml` file.
2. Edit the arguments element in the `cm-admin.xml` file to look like the following:

```
<service>
  <id>cm-admin</id>
  <name>Connection Manager Admin</name>
  <description>This service runs CM-ADMIN system.</description>
  <executable>java</executable>
  <arguments>-DTYPE=MVCM -jar
     -DAWS_ACCESS_KEY=<AWS_Access_Key>
     -DAWS_SECRET_KEY=<AWS_Secret_Key>
     -DAWS_REGION=<AWS_Region>
     -DBUCKET_NAME=<bucket_name>
     -DCONFIG_NAME=<main_config_file_name>
   "%BASE%\
  cm-admin.jar"</arguments>
  <logmode>rotate</logmode>
  <logpath>%BASE%\logs</logpath>
```

3. Restart the MVIS Admin service for your changes to take effect.

Parent topic: Configuring MVIS and the MVIS Admin for Amazon S3 storage

Chapter 6: Enabling security for MVIS

To enable security for MVIS, complete the following tasks:

- Encrypting the password for the MVIS Web Admin UI and MVIS Admin
  The MVIS Admin and MVIS Web Admin UI are protected by credentials that are stored in the `application.properties` configuration file. By default, the user and password are both `admin`. These can be changed by editing the `application.properties` file and encrypting the password.
- Setting up SSL
  The following three areas of MVIS communication can be secured using SSL:
- REST server security configuration
  You can create HTTP users and define access controls to secure specific subroutines and data resources or an entire REST server account. Once secured, users will need to provide valid credentials to access the subroutine, data resource or REST server.

Encrypting the password for the MVIS Web Admin UI and MVIS Admin

The MVIS Admin and MVIS Web Admin UI are protected by credentials that are stored in the `application.properties` configuration file. By default, the user and password are both `admin`. These can be changed by editing the `application.properties` file and encrypting the password.

About this task

The MVIS Admin and the MVIS Web Admin UI are protected with HTTP BASIC authentication. The credentials are stored and encrypted in the `application.properties` file under the **[Auth]** section. The user name is stored under the auth.user property and the encrypted password is stored under the auth.password property, as shown in the following example:

```
application.properties
...
[Auth]
auth.user=admin
auth.password=TdH7VWfzmuie9cFPVLHlKQ==
...
```

Note, without the -i, clear characters are echoed on the screen. When a carriage return is enter, the encrypted password is shown.

Procedure

To encrypt the password, use the `hazify` command, as shown is the following example:

```
c:\u2\cm> java -jar hazify.jar -i
Enter password: <type password and press enter>
Enter password again: <type password and press enter>
Encrypted password is: SBDwStnkNF5WSmMH0oCrfQ==
```

Parent topic: Enabling security for MVIS

Setting up SSL

The following three areas of MVIS communication can be secured using SSL:

- Security between the client and the MVIS Admin
  The browser requires the signing certificate in its trust store. Generally, this is not necessary, as most common CA certificates are available in your truststore. If you are using self-signed certificates, you will need to install your certificate in your trusted root certificate authorities store.
- Security between applications and MVIS
  Secure your application and MVIS using the `cm.ini` file.
- Security between MVIS and the database
  To secure the connection between your data server and MVIS, you must first configure your database connection with the MultiValue Integration Server for SSL. This entails creating or obtaining a certificate for client and one for the server. The signing certificate should be placed in your truststore on the MVIS side. The server certificate is referenced by the database. See either your UniVerse or UniData *Security Features* guide for additional instructions on creating a certificate on the database.

Parent topic: Enabling security for MVIS

Security between the client and the MVIS Admin

The browser requires the signing certificate in its trust store. Generally, this is not necessary, as most common CA certificates are available in your truststore. If you are using self-signed certificates, you will need to install your certificate in your trusted root certificate authorities store.

Procedure

1. Obtain a valid certificate and private key to secure the connection between the MVIS Admin and your web browser. Specific instructions for creating or obtaining a certificate are outside the scope of this documentation.
2. Place the certificate and the private key from Step 1 into your keystore.
3. In the **[SSL]** section of the `application.properties` file, make the following changes to enable an HTTPS connection between the MVIS Admin and the client:
   a. In the **server.port** field, designate the server port to be used for secure access to the MVIS Admin. The default is 7043.
   b. In the **server.ssl.key-store** field, enter the path to the keystore referenced in Step 2.
   c. In the **server.ssl.key-store-password** field, enter the password for your keystore.
   d. In the **server.ssl.key-password**, enter the password you obtained in Step 1.
4. Save the `application.properties` file.
5. If the MVIS Admin is already running, you must restart it to make these changes effective.
6. Verify that your secure connection is working by opening your web browser and entering `https://`*`MVIS Admin`*`:`*`port`*, for example: `https://localhost:7043`.
   where:
   - *MVIS Admin* is the DNS name or IP address of the server where the MVIS Admin is running.
   - *port* is the server port number specified for the **server.port** field in step 3a.

Parent topic: Setting up SSL

Security between applications and MVIS

Secure your application and MVIS using the `cm.ini` file.

Prerequisites

You need a certificate, and you can use the same one created in Security between the client and the MVIS Admin, on page 43.

Procedure

1. Open the `cm.ini` file in a text editor.
   a. In the **[Default]** section, set **uoSSLEnabled** to `true`.
   b. Add the sslKeyStorePath= property and append the path to your keystore. For example, `sslKeyStorePath=C:\certs\den-tci009.jks`
   c. Add the sslKeyStorePassword= property and append the password for your keystore, for example `sslKeyStorePassword=abcd`.
2. Save the `cm.ini` file.
3. 3. If MVIS is running, restart MVIS to make your changes effective.

Parent topic: Setting up SSL

Security between MVIS and the database

To secure the connection between your data server and MVIS, you must first configure your database connection with the MultiValue Integration Server for SSL. This entails creating or obtaining a certificate for client and one for the server. The signing certificate should be placed in your truststore on the MVIS side. The server certificate is referenced by the database. See either your UniVerse or UniData *Security Features* guide for additional instructions on creating a certificate on the database.

Prerequisites

- Add an account.
- Have the root.cer certificate for the client side and the server certificate signed by root.cer.
- Create a security context record and configure the data server using XAdmin.

For information on creating a Security Context Record and configuring the data server with XAdmin, see the *Rocket DBTools User Guide*.

About this task

Note: To ensure secure only connections from MVIS to the database, you must define a secure flag for that Unirpc port

Procedure

1. From the MVIS Web Admin UI, select **Configuration** → **Security**, and complete the following steps:
   a. In the **Use Trust Store** field, select **Enabled**.
   b. In the **SSL Trust Store Path** field, enter the path to the truststore where your client certificate is located.
   c. In the **SSL Trust Store Password** field, enter the password for the truststore

2. From the MVIS Web Admin UI, select **Configuration** → **Accounts**, and then click the **Edit** icon.
3. From the **Server details** tab, complete the following fields:

Field	Action
SSL connection(s) to MV server	Select this check box to activate SSL mode for the account.
SSL hostname validation	Select this check box to compare the server name in the account configuration to the server name in the MVIS property that is used for the server-side certificate.

4. Select one of the following options:
   - To start MVIS, in the **MultiValue Integration Server Status** section, click **Start**.
   - To restart your account, from the **Accounts** section, click **Restart Account**.

   If the server name in the `cm.ini` file and the server certificate match, your server name connection pools for the account are created successfully. If the sever name and server certificate do not match, a `Bad connection...` error is displayed.

Parent topic: Setting up SSL

REST server security configuration

You can create HTTP users and define access controls to secure specific subroutines and data resources or an entire REST server account. Once secured, users will need to provide valid credentials to access the subroutine, data resource or REST server.

To enable security for a REST server, complete the following tasks:

- Defining HTTP users
  You can create HTTP users and assign them to specific roles to secure access to REST subroutines, data resources or the REST server itself. Complete these steps to define a new http user.
- Defining access control
  Set up MVIS to secure specific subroutines and data resources or an entire REST server account. Once secured, users will need to provide valid credentials to access the subroutine, data resource or REST server.

Parent topic: Enabling security for MVIS

Defining HTTP users

You can create HTTP users and assign them to specific roles to secure access to REST subroutines, data resources or the REST server itself. Complete these steps to define a new http user.

To enable authentication for REST services running under MVIS, select **Configuration** from the **Administration** menu, and open the **Security** section. In the **HTTP Authentication** drop down, select either **http-basic** or **http-digest** authentication. After making your selection, restart MVIS for your changes to take effect.

1. From the MVIS Web Admin UI, select **Administration** → **REST Server**, and then expand the **REST Server Security Configuration** section.
2. Expand the **Define HTTP Users** section.
3. Click the **Add HTTP User** button.

4. On the Define HTTP Users dialog box, enter a user name and password for the HTTP user.

   Note: The name and password entries are independent from user names and passwords stored in other configuration files in this product.

5. Choose a role for the new user from the **Roles** list by selecting the appropriate check box. Note that a user can have multiple roles.

   Tip: To create a new role, click **Add Role**, type a name for the new role and select the role's check box to apply the role to the new HTTP user.

6. Click **OK**.

   The new HTTP user is added to the list of defined HTTP users.

Parent topic: REST server security configuration

Defining access control

Set up MVIS to secure specific subroutines and data resources or an entire REST server account. Once secured, users will need to provide valid credentials to access the subroutine, data resource or REST server.

1. From the MVIS Web Admin UI, select **Administration** → **REST Server**, and then expand the **REST Server Security Configuration** section.
2. Expand the **Define Access Control** section.
3. Click the **Add Path** button.
4. In the Define Access Control Path dialog box, select the account or specific subroutine or data resource to secure from the **Path** drop-down and then click **OK**.
5. In the **Define Access Control** section, click the **Roles and Methods** drop-down and select the **GET**, **POST**, **PUT** and **DELETE** methods to allow for users assigned to any of the roles listed in the **Any Role** column.

   Tip:

   - Select the top-most check box for any of the **GET**, **POST**, **PUT** or **DELETE** methods to allow that method for all defined roles.
   - To create a new role, click **Add Role**, type a name for the new role and select the **GET**, **POST**, **PUT** or **DELETE** methods to require that a logged-in user be a member of the specified group to be granted access to the end point(s). Note that your changes are immediately saved.

6. Restart MVIS for your changes to take effect.

Parent topic: REST server security configuration

Appendix A: cm.ini file properties

Use the `cm.ini` file to configure MVIS.

[LogLevel] section

The **[LogLevel]** section of the `cm.ini` file contains parameters for the types of information to write to the `cm.log` file. The following example shows the **[LogLevel]** section:

```
[LogLevel]
logging.level=ERROR
```

The following table lists the available logging levels. Each level, with the exception of ****, enables the logging level selected and every logging level beneath the selected level should be sent to the `cm.log` file. For example, selecting the Logging level **INFO** includes logging at the **INFO**, **WARNING**, and **ERROR** logging levels. Selecting **DEBUG** includes **DEBUG**, **INFO**, **WARNING**, and **ERROR** logging levels.

Logging level	Description
ERROR	Logs errors only.
WARNING	Logs warnings and errors.
INFO	Logs informational messages and above.
DEBUG	Logs debug messages and above.
TRACE	Logs everything.
OFF	Disables Logging.

[Default] section

The following table describes the properties in the **[Default]** section:

Property	Description
connectionManagerPort	The port number on which MVIS accepts UO requests. The default is 7870.
connectionTracingEnabled	If set to **true**, detailed connection activity from MVIS to the data servers is logged.
excludedCipherSuites	Specifies the list of cipher suites that should be rejected by MVIS. The cipher suites should be delimited by a space. The default is empty.
excludedProtocols	Specifies the list of SSL protocols that should be rejected by MVIS. The protocols should be delimited by a space. The default is empty.
gracefulShutdownEnabled	Specifies whether the http server should be given time to gracefully shutdown before stopping MVIS. The default is **false**.
gracefulShutdownTimeout	The timeout to use for the graceful shutdown of the http server, in milliseconds, if **gracefulShutdownEnabled** is set to true. The default value (if enabled) is 30000 milliseconds (30 seconds).
httpAuthentication	Specifies authentication to be used for the http server. Can be http-basic, http-digest, or none.
httpIdleTimeout	Number milliseconds before an idle http connection from a REST client times out. The default is 30000 (30 seconds).

Property	Description
httpsEnabled	Specifies whether to use SSL protocol on the http port **httpPort**. The default is **false**.
httpPort	The port to use for the http server. The default is 7171.
idleConnectionCheckInterval	The number of milliseconds to wait before checking for and removing backend data server connections whose idle time has exceeded the **idleConnectionTimeout**. The default is 60000 milliseconds (60 seconds).
idleConnectionTimeout	The number of milliseconds that a connection to a backend data server can remain idle before it is flagged for removal. The default is 60000 milliseconds (60 seconds).
logPath	The full path to the main log file (e.g. `C:\u2\cm\logs\cm.log` or `/opt/cm/logs/cm.log`.
maxPoolRestartWaitTime	Specifies the maximum times in second to retry the restarting or deletion of an account, if connections in the account are found to be busy when the restart or deletion is requested. The default is 60 seconds.
monitorPort	The port number on which MVIS accepts monitor and maintenance requests made from the MVIS Admin. The default is 7871.
openSessionTimeout	For a given account, specifies the maximum amount of time to wait for a response from a data server connection before timing out. Default is 300 seconds. It is important to note that when MVIS is unable to connect to the server, a second attempt to connect will automatically be made before suspending further attempts. This means that a **openSessionTimeout** of 10 seconds will effectively take 20 seconds before timing out. Clients who utilize this timeout in multi-tier environments where UniRPC sits between the server and MVIS should take this behavior into account when setting up their **openSessionTimeout** value.
restMaxRequestBodySize	The maximum size in bytes of a request payload for rest endpoints. The default is 200000 (200K bytes).
servicesCacheSize	The number of services that will be cached by the running MVIS after they are loaded from disk or other configuration storage. Services are grouped together based on their definition file, and cached after they are loaded. **servicesCacheSize** dictates how many of such service definition files are cached. A **servicesCacheSize** of 0 indicates no caching will be done. This is useful during development, but can affect performance as all services files will need to be loaded with each service-based request. If service files are stored in cloud storage, this will require a network round-trip. The default is 50.
sslKeyManagerPassword	Optional use by the Jetty http server in MVIS – the password (if any) for the specific key within the key store.
sslKeyStorePassword	The password to the Java keystore specified in **sslKeyStorePath**. The password should be encrypted.
sslKeyStorePath	The path to the Java keystore.
sslTrustStorePassword	If **trustStoreEnabled** is set to **true**, specifies the password to the Java truststore specified in **sslTrustStorePath**. The password should be encrypted.

Property	Description
sslTrustStorePath	If **trustStoreEnabled** is set to **true**, specifies the path to a Java truststore to be used. If **trustStoreEnabled** is not set to **true**, this property is ignored and the default JVM truststore will be used.
startPerfStatsLoggingEnabled	Specifies whether to log performance statistics. Performance statistics are written to the log file specified in **statisticsLogPath**. The default is **false**.
trustStoreEnabled	If set to **true**, this property specifies that the truststore specified in **sslTrustStorePath** be used, rather than the JVM's default truststore. The default is **false**.
uoClientConnectionTimeout	Specifies the number of milliseconds that sockets servicing UO clients can be left open with no activity before they timeout on the listening thread. The default is 28800 (8 hours).
uoMaxServerRequestQueue	Number of requests that can be queued on the listening socket when accepting connections from UO clients. The default value is 50 requests.
uoSSLEnabled	Whether to use SSL protocol on the **connectionManagerPort**, for UO clients connecting to MVIS. The default is **false**.

[Account] section

The following table describes the **[Account]** section properties:

Property	Description
accountEnabled	If set to true (default), enables the account in the cm.ini file. If set to false, disables the account in the cm.ini file.
accountpath	The path to the MV account on the data server.
avgRespTimeThresholds	Triggers monitor warnings when the overall average request response time for this account exceeds these values. The average covers the period since MVIS was last started. Two space-delimited values are required for this property. The first value is the amount of time in milliseconds which will cause the monitor to display warning status. The second value is the amount of time in milliseconds which will cause the monitor to display critical status.
connectionString	The entry from the **unirpcservices** file on the system where the specified account resides, used when connecting to the account.
devModeEnabled	For a given account, setting this value allows all requests to get the latest version of compiled code, instead of using a cached version of the object code. **devModeEnabled** causes database processes to restart after each method call. While allowing newly compiled code to be picked up, this mode carries a performance overhead as the associated database processes terminate and are recreated after each method call, and is not recommended for live production use for this reason. The default is **false**.
encoding	For a given account, specifies the encoding for the pooled connections.

Property	Description
executionTimeout	For a given account, specifies the maximum amount of time to wait for a response from a data server connection before timing out. Default is 300 seconds.
maxPoolSize	The maximum number of connections maintained in the connection pool for a given account.
minPoolSize	The minimum number of connections maintained in the connection pool for a given account.
mvServer	The name or IP address of the data server where the account is located.
mvServerConnectionTimeout	For a given account, specifies the maximum amount of time to wait for a response from a data server connection before timing out. Default is 300 seconds.
mvServerKeepAliveEnabled	For a given account, specifies if a **mvServerKeepAliveEnabled** ping should be sent at a recurring interval. Enabling this allows MVIS to actively check for problematic connections to the data servers. This active ping allows MVIS to detect a wider class of network-related problems. If MVIS detects a problem with a connection in its pool, it will attempt to replace the problematic connection.
mvServerKeepAliveInterval	For a given account, specifies the number of milliseconds to wait between keep alive ping requests for the account. The default, if **mvServerKeepAliveEnabled** is **true**, is 60000 (60 seconds).
mvServerPort	For a given account, specifies the port on the data server to which MVIS will connect. The default is 31438
password	For a given account, specifies the password to the operating system-level user ID that will be used to connect to the data server. The password should be encrypted.
protocol	Specifies the protocol that this account should use when being accessed via a client. ▪ **UNIRPC:** Use UO protocol for this account (default). ▪ **REST:** Use http for this account.
requestsQueuedThresholds	Triggers monitor warnings when the number of requests for this account waiting to be served exceeds these values. Two space-delimited values are required for this property. The first value is the number of requests which will cause the monitor to display warning status. The second value is the number of requests which will cause the monitor to display critical status.
respTimeThresholds	Triggers monitor warnings when the average response time for this account exceeds these values. The average covers a single monitor refresh interval. Two space-delimited values are required for this property. The first value is the amount of time in milliseconds which will cause the monitor to display warning status. The second value is the amount of time in milliseconds which will cause the monitor to display critical status.
sbPassword	The password for the SB+ user to use in REST calls that require an SB+ environment.

Property	Description
sbSysId	The name of the SB+ system id to use in REST calls that require an SB+ environment.
sbUserId	The SB+ user id to use in REST calls that require an SB+ environment.
serverSideLoggingEnabled	Enables como files for the account. The default is **false**.
slowRequestThreshold	Triggers monitor warnings when any requests exceed this value for the given account.
soapPort	The port number on which MVIS accepts SOAP requests for this account.
soapTimeout	For a given account, specifies the timeout, in milliseconds, for the SOAP worker thread when it reads data from the socket. The default is 5000.
sslHostnameValidationEnabled	When set to **true**, this specifies that the SSL connections, from MVIS to the data server, for the account require the hostname on the data server's SSL certificate match the server name for the account. By default, this is set to **false**.
uoClientTCPKeepAliveEnabled	Enables the TCP keep alive option on UO client requests for this account into MVIS. The default is **false**.
uoClientValidationEnabled	Validates credentials passed in through UO clients, unless set to **false**. The default is **false**.
uoSSLEnabled	Specifies whether to use the SSL protocol when communicating with the data server, for this account. The default is **false**.
userId	For a given account, specifies the operating system-level user ID that will be used to connect to the data server.

Appendix B: MVIS Java options

Use Java options to supply configuration parameters on the command line to either MVIS or the MVIS Admin.

> **Note:** Any property in the `application.properties` file can be overridden on the command line by specifying the equivalent Java option for the MVIS Admin. For example, to override the auth.user property found in the `application.properties` file, use the –`Dauth.user=<user_name>` Java option.

Cloud

The following table describes the Java properties that you can use to configure cloud environments:

Property	Description
-DAWS_ACCESS_KEY=<AWS_Access_Key>	AWS access key
-DAWS_SECRET_KEY=<AWS_Secret_Key>	AWS secret key
-DAWS_REGION=<AWS_Region>	AWS region name
-DAZURE_ACCOUNT=<Azure_Storage_Account>	Azure storage account name
-DAZURE_ACCOUNT_KEY=<Azure_Storage_Account_Key>	Azure storage account key
-DAZURE_SB_NAMESPACE<Azure_Service_Bus_Namespace>	Azure service bus namespace
-DAZURE_SB_SAS_KEY=<Azure_Service_Bus_Shared_Access_Signature_Key>	Azure service bus shared access signature key
-DBUCKET_NAME=<container_name>	Name of the container storage to hold the MVIS configuration files.

Miscellaneous

The following table describes the Java properties that is used miscellaneously:

Property	Description
-DTYPE=MVCM	(always required to be MVCM)

Configuration

The following table describes the Java properties that you can use to configure MVIS and the MVIS Admin:

Property	Description
-DFS_DIR=<file_system_directory_name_for_config_files>	When MVIS and the MVIS Admin are reading from the file system, use FS_DIR to specify what directory you can find the configuration files.

Property	Description
-DCONFIG_NAME= <main_config_file_name>	Name of the main MVIS configuration file.
-DACCOUNTS_CONFIG_NAME= <accounts_config_file_name>	MVIS can optionally store its account sections in a separate configuration file from the main configuration file.

Logging

The following table describes the Java properties that you can use to configure logging:

Property	Description
-DCLOUD_LOGGING=AZURE|AWS	Specifies logs to be sent to Azure Application Insights or AWS CloudWatch.
-DAPPLICATION_INSIGHTS_IKEY= <Azure_Application_Insights_Key>	Required for Application Insights. If set, and performance statistics logging is turned on in the MVIS Web Admin UI, performance statistics will be sent to Application Insights.
-DAWS_LOG_NAME=<AWS_Log_Group_Name>	AWS log group name
-DAWS_LOG_STREAM=<AWS_Log_Stream_Name>	AWS log stream name
-DCONSOLE_LOGGING=1	Sends all log data to the console and any additional log targets.
-DLOG_LEVEL=ERROR| WARN|INFO|TRACE|DEBUG|OFF	Specify the starting log level when writing to the `startCM.log` file. The `startCM.log` file is a bootstrap log file that is written to when MVIS first starts and before the main log file has been set in the configuration file.
-DROLLOVER_ON_START=1	Roll over the log files each time MVIS starts

Appendix C: Logging and monitoring with fluentd

MVIS logging capabilities fully integrate with fluentd allowing you to use your logging and monitoring solution of choice (for example, DataDog, Splunk, or Prometheus).

Fluentd is an open source data collector that allows you to implement a unified logging scheme for collecting, filtering, buffering and outputting data logs across multiple sources and destinations. Fluentd's architecture allows you to extend MVIS functionality through hundreds of plugins that connect dozens of data sources and data outputs.

Fluentd can be used with MVIS in a Container format or without containers. To run it in non-container format, separately install fluentd, and run it with the fluentd `cm config` file. To deploy fluentd and MVIS in a container, follow the examples below.

Note: To use fluentd with DataDog, Splunk, or Prometheus, see the `Fluentd_logging_examples` folder of your installation download. General prerequisites for building a solution with any plugin is as follows:

- Create a docker image with the required plugin (docker images for DataDog, Splunk, Prometheus are attached)
- Create a K8s cluster (or install MVIS, MVIS Admin, fluentd to your local machine
- Deploy MVIS, MVIS Admin, Redis and use the specific fluent configmap

---

Prerequisites

- AWS keys
- Docker
- kubectl
- K8S cluster

Example of building a docker container with Fluentd plugins for AWS CloudWatch

Docker file

```
FROM fluent/fluentd:v1.3-onbuild
MAINTAINER YOUR_NAME user@company.com
RUN apk add --no-cache --update --virtual
.build-deps sudo build-base
ruby-dev \
# cutomize following instruction as you wish
&& sudo gem install fluent-plugin-cloudwatch-logs \
&& sudo gem sources --clear-all \
&& apk del .build-deps \
&& rm -rf /home/fluent/.gem/ruby/2.3.0/cache/*.gem

EXPOSE 24284
```

Build docker image

```
docker build -t ecr-url-cm-cloud-logs:latest ./
```

Push the docker image to the ECR service to be able to use this image in K8S

```
docker push ecr-url-cm-cloud-logs:latest ./
```

Examples for creating the k8s MVIS, MVIS Admin and Redis services

Redis

Config Map

```
apiVersion: v1
kind: Service
metadata:
  name: redis
spec:
  #type: LoadBalancer
  ports:
    - port: 6379
      name: redis
  selector:
    app: redis
---
apiVersion: apps/v1beta2
kind: StatefulSet
metadata:
  name: redis
spec:
  selector:
    matchLabels:
      app: redis  # has to match .spec.template.metadata.labels
  serviceName: redis
  replicas: 1
  template:
    metadata:
      labels:
        app: redis  # has to match .spec.selector.matchLabels
    spec:
      containers:
        - name: redis
          image: redis:3.2-alpine
          imagePullPolicy: IfNotPresent
          #args: ["--requirepass", "$(REDIS_PASS)"]
          args: ["--appendonly", "yes", "--save", "900", "1", "--save", "30", "2"]
          ports:
            - containerPort: 6379
              name: redis
```

Apply

```
kubectl apply -f redis.yaml
```

Config Map

Config Map

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: fluentd-config
  labels:
    k8s-app: fluentd
data:
```

```
  fluent.conf: |
    <source>
          @type tail
          tag cm-admin.log
          path /opt/cm/logs/cm-admin.log
          pos_file /opt/cm/logs/cm-admin.log.pos
          format multiline
          format_firstline /\d{4}-\d{1,2}-\d{1,2}/
          format1 /^(?<time>\d{4}-\d{1,2}-\d{1,2} \d{1,2}:\d{1,2}:\d{1,2}:\d{1,3}) \[(?<l
    </source>

    <filter cm-admin.log>
      @type record_transformer
      <record>
        host_param "#{Socket.gethostname}"
      </record>
    </filter>

    <match cm-admin.log>
        @type cloudwatch_logs
        log_group_name cm-admin
        log_stream_name adminlog
        auto_create_stream true
    </match>
```

Apply

```
kubectl apply -f configmap.yaml
```

MVIS Admin

CM Admin YAML

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: cmadmin
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: cmadmin
    spec:
      imagePullSecrets:
      - name: cmsecret
      containers:
      - name: cmadmin
        image: 000578147072.dkr.ecr.us-west-2.amazonaws.com/cmadmin:latest
        ports:
        - containerPort: 7077
        env:
        - name: JAVA_OPTS
          value: -DTYPE=MVCM -DCONSOLE_LOGGING=1 -DREDIS_HOST=redis
        volumeMounts:
        - name: cmlogs
          mountPath: /opt/cm/logs
      - name: fluentd
        image: 000578147072.dkr.ecr.us-west-2.amazonaws.com/fluentd:cloudwatch
        imagePullPolicy: Always
        ports:
          - containerPort: 2020
```

```
        env:
        - name: AWS_REGION
          value: "us-west-2"
        - name: AWS_ACCESS_KEY_ID
          value: "xxx"
        - name: AWS_SECRET_ACCESS_KEY
          value: "xxxxx"
        volumeMounts:
        - name: cmlogs
          mountPath: /opt/cm/logs
        - name: fluentd-config
          mountPath: /fluentd/etc/
      volumes:
        - name: cmlogs
          emptyDir: {}
        - name: fluentd-config
          configMap:
            name: fluentd-config
---
apiVersion: v1
kind: Service
metadata:
  name: cmadmin
spec:
#  type: LoadBalancer
  ports:
  - port: 7077
  selector:
    app: cmadmin
```

Apply

```
kubectl apply -f cm-admin.yaml
```

MVIS

CM YAML

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: cm
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: cm
    spec:
      imagePullSecrets:
      - name: cmsecret
      containers:
      - name: cm
        image: 000578147072.dkr.ecr.us-west-2.amazonaws.com/cm:latest
        ports:
        - containerPort: 7071
        - containerPort: 7171
        - containerPort: 7070
        env:
        - name: JAVA_OPTS
          value: "-DTYPE=MVCM -DCONSOLE_LOGGING=1 -DREDIS_HOST=redis"
     - name: fluentd
```

```
          image: 000578147072.dkr.ecr.us-west-2.amazonaws.com/fluentd:cloudwatch
          imagePullPolicy: Always
          ports:
            - containerPort: 2020
          env:
          - name: AWS_REGION
            value: "us-west-2"
          - name: AWS_ACCESS_KEY_ID
            value: "xxx"
          - name: AWS_SECRET_ACCESS_KEY
            value: "xxxxx"
          volumeMounts:
          - name: cmlogs
            mountPath: /opt/cm/logs
          - name: fluentd-config
            mountPath: /fluentd/etc/
        volumes:
          - name: cmlogs
            emptyDir: {}
          - name: fluentd-config
            configMap:
              name: fluentd-config
---
apiVersion: v1
kind: Service
metadata:
  name: cm
spec:
  # type: LoadBalancer
  ports:
  - port: 7070
    name: cmport
  - port: 7171
    name: cmrest
  - port: 7071
    name: cmmonitor
  selector:
    app: cm
```

Apply

```
kubectl apply -f cm.yaml
```

Index