Red Hat JBoss Data Virtualization 6.3
Virtual Database Builder Guide

Learn how to use the new CLI-based VDB Builder tool.

David Sage

Red Hat JBoss Data Virtualization 6.3 Virtual Database Builder Guide

Learn how to use the new CLI-based VDB Builder tool.

David Sage
dlesage@redhat.com

Legal Notice

Abstract

Table of Contents

PREFACE

CHAPTER 1. GOALS OF THIS GUIDE

Warning

This is a technology preview only. Technology Preview features are not supported, may not be functionally complete, and are not intended for production use. These features are included to provide customers with early access to upcoming product innovations, enabling them to test functionality and provide feedback during the development process.

The VDB Builder provides a command line interface which allows the user to create and edit virtual databases (VDBs) and other artifacts from the command line. This guide aims to teach you the tool quickly by showing you some examples that will help you perform key tasks with it.

CHAPTER 2. CREATE A VDB INTERACTIVELY

This sample shows how to use the VDB Builder command line interface to create a new dynamic VDB interactively. Use this sample as a starting point to create your own, more complex VDBs.

Note

You can use **tab completion** to see the available commands options, or use **help commandName** to see command details.

Figure 2.1. Code sample

```
[workspace] list
     There are no children for workspace '/workspace'.
[workspace] create-vdb SampleVDB
     VDB 'SampleVDB' was created.
[workspace] cd SampleVDB
[SampleVDB (VDB)] add-model Model1
     Model 'Model1' was successfully added.
[SampleVDB (VDB)] cd Model1
[Model1 (Model)] add-table Table1
     Table 'Table1' was successfully added.
[Model1 (Model)] cd Table1
[Table1 (Table)] show-summary
     Properties for Table '/workspace/SampleVDB/Model1/Table1':

     NAME                          VALUE                         DEFAULT
     ------------------------      ------------------------      ------------------------
     ANNOTATION                    <no value>
     CARDINALITY                   <no value>                    -1
     MATERIALIZED                  <no value>                    false
     MATERIALIZED_TABLE            <no value>
     NAMEINSOURCE                  <no value>
     UPDATABLE                     <no value>                    true
     UUID                          <no value>
     onCommitValue                 <no value>
     queryExpression               <no value>
     schemaElementType             FOREIGN                       FOREIGN
     temporary                     <no value>

     There are no children for Table '/workspace/SampleVDB/Model1/Table1'.
[Table1 (Table)] add-column Col1
     Column 'Col1' was successfully added.
[Table1 (Table)] add-column Col2
     Column 'Col2' was successfully added.
[Table1 (Table)] cd Col1
[Col1 (Column)] set-property datatypeName string
     The 'datatypeName' property was successfully set.
[Col1 (Column)] cd ../Col2
[Col2 (Column)] set-property datatypeName string
     The 'datatypeName' property was successfully set.
[Col2 (Column)] cd ..
[Table1 (Table)] show-children
     Children for Table '/workspace/SampleVDB/Model1/Table1':

     NAME                          TYPE
     ------------------------      ------------------------
     Col1                          Column
     Col2                          Column
[Table1 (Table)] workspace
[workspace] list
     Children for workspace '/workspace':

     NAME                          TYPE
     ------------------------      ------------------------
     SampleVDB                     VDB
[workspace]
```

Explanation of the Commands Used in this Example

list

shows all children at the CLI context. (The workspace in this example initially contains no children.)

create-vdb SampleVDB

create a VDB named SampleVDB.

cd SampleVDB

change directory to go into the SampleVDB.

add-model Model1

add a model named Model1 to the VDB.

cd Model1

navigate into the Model1 CLI context.

add-table Table1

add a table named Table1 within Model1.

cd Table1

navigate into the Table1 CLI context.

show-summary

shows a summary of the current cli context, (in this case it is Table1). Show summary will show the object properties, as well as its children.

add-column Col1

add column named Col1 within Table1.

add-column Col2

add column named Col2 within Table1.

cd Col1

navigates into column Col1.

set-property datatypeName string

this sets the Col1 datatype to string.

cd ../Col2

navigates up a level, then down into the Col2 column.

set-property datatypeName string

sets Col2 datatype to string.

cd ..

navigates up one level.

show-children

show the children of the current context Table1. Notice the child columns that were created.

workspace

navigates to the workspace root context.

list

list the workspace children. (It now contains the SampleVDB.)

CHAPTER 3. IMPORT AND EDIT A VDB INTERACTIVELY

This sample shows you how to import the SampleVDB and how to then edit it to set a property (in this case the VDB version).

Figure 3.1. Code sample

```
[workspace] list
     There are no children for workspace '/workspace'.
[workspace] upload-vdb SampleVDB ./SampleVDB-vdb.xml
     VDB 'SampleVDB' was successfully uploaded.
[workspace] list
     Children for workspace '/workspace':

     NAME                          TYPE
     ------------------------      ------------------------
     SampleVDB                     VDB
[workspace] cd SampleVDB
[SampleVDB (VDB)] show-properties
     Properties for VDB '/workspace/SampleVDB':

     NAME                          VALUE                                 DEFAULT
     ------------------------      ----------------------------------    --------------------------------
     allowed-languages             <no value>
     authentication-type           <no value>
     connectionType                BY_VERSION
     description                   <no value>
     gss-pattern                   <no value>
     name                          SampleVDB
     originalFile                  /tko:komodo/tko:workspace/SampleVDB
     password-pattern              <no value>
     preview                       false                                 false
     query-timeout                 <no value>
     security-domain               <no value>
     version                       1                                     1
[SampleVDB (VDB)] set-property version 2
     The 'version' property was successfully set.
[SampleVDB (VDB)] show-properties
     Properties for VDB '/workspace/SampleVDB':

     NAME                          VALUE                                 DEFAULT
     ------------------------      ----------------------------------    --------------------------------
     allowed-languages             <no value>
     authentication-type           <no value>
     connectionType                BY_VERSION
     description                   <no value>
     gss-pattern                   <no value>
     name                          SampleVDB
     originalFile                  /tko:komodo/tko:workspace/SampleVDB
     password-pattern              <no value>
     preview                       false                                 false
     query-timeout                 <no value>
     security-domain               <no value>
     version                       2                                     1
[SampleVDB (VDB)] workspace
[workspace] list
     Children for workspace '/workspace':

     NAME                          TYPE
     ------------------------      ------------------------
     SampleVDB                     VDB
[workspace]
```

Explanation of the Commands Used in this Example

list

shows all children at the CLI workspace context. (The workspace in this example initially contains no children.)

upload-vdb SampleVDB ./SampleVDB-vdb.xml

upload the SampleVDB-vdb.xml file found in the vdbbuilder directory.

list

list the workspace children again (it now contains the SampleVDB).

cd SampleVDB

change into the SampleVDB directory.

show-properties

shows the VDB’s properties.

set-property version 2

sets the version to '2'.

show-properties

shows the VDB properties again. Note that the version is now '2'.

workspace

navigate to the workspace root context.

list

list the workspace children.

CHAPTER 4. IMPORT AND EXPORT VDBS AND DDL

This example teaches you how to import and export a dynamic VDB. The session shows the importation of PartsVDB-vdb.xml, how to change the VDB version, and how to export of the VDB into a different dynamic VDB file.

Figure 4.1. Code sample

```
[workspace] list
    There are no children for workspace '/workspace'.
[workspace] upload-vdb PartsVDB /home/mdrilling/Desktop/TestVDBs/PartsVDB-vdb.xml
    VDB 'PartsVDB' was successfully uploaded.
[workspace] list
    Children for workspace '/workspace':

    NAME                          TYPE
    ------------------------      ------------------------
    PartsVDB                      VDB
[workspace] cd PartsVDB
[PartsVDB (VDB)] show-properties
    Properties for VDB '/workspace/PartsVDB':

    NAME                          VALUE                                DEFAULT
    ------------------------      ---------------------------------    --------------------------------
    allowed-languages             <no value>
    authentication-type           <no value>
    connectionType                BY_VERSION
    description                   Sample Parts Supplier example VDB
    gss-pattern                   <no value>
    name                          PartsVDB
    originalFile                  /tko:komodo/tko:workspace/PartsVDB
    password-pattern              <no value>
    preview                       false                                false
    query-timeout                 <no value>
    security-domain               <no value>
    validationDateTime            Mon Jul 06 14:15:43 CDT 2015
    validationVersion             8.7.1
    version                       1                                    1
[PartsVDB (VDB)] set-property version 2
    The 'version' property was successfully set.
[PartsVDB (VDB)] export-vdb ./ExportedPartsVDB-vdb.xml
    VDB 'PartsVDB' was successfully exported to './ExportedPartsVDB-vdb.xml' with override flag of 'false'.
[PartsVDB (VDB)]
```

This is how you upload a VDB as "vdbName" from "vdbFile" into the workspace:

```
upload-vdb <vdbName> <vdbFile>
```

This is how you export the current context "PartsVDB" to a specified file called "vdbFile":

```
export-vdb <vdbFile>
```

This example shows you how to import and export DDL. It depicts the creation of a VDB called MyVDB, the importation of Teiid-MySQLAccounts.ddl which is undertaken in order to create the content for Model1, and then the export of the model’s contents into a different DDL file:

Figure 4.2. Code sample

```
[workspace]
[workspace] create-vdb MyVDB
     VDB 'MyVDB' was created.
[workspace] cd MyVDB
[MyVDB (VDB)] upload-model Model1 PHYSICAL /home/mdrilling/Desktop/TestDDL/Teiid-MySQLAccounts.ddl
     Model 'Model1' was successfully uploaded.
[MyVDB (VDB)] list
     Children for VDB '/workspace/MyVDB':

     NAME                          TYPE
     ------------------------      ------------------------
     Model1                        Model
[MyVDB (VDB)] cd Model1
[Model1 (Model)] list
     Children for Model '/workspace/MyVDB/Model1':

     NAME                          TYPE
     ------------------------      ------------------------
     accounts.ACCOUNT              Table
     accounts.CUSTOMER             Table
     accounts.HOLDINGS             Table
     accounts.PRODUCT              Table
     accounts.SUBSCRIPTIONS        Table
[Model1 (Model)] export-ddl ./ExportedModel1.ddl
     Model 'Model1' was successfully exported to './ExportedModel1.ddl' with override flag of 'false'
[Model1 (Model)]
```

This is how you create a model of "modelType" called "modelName", using the ddl in "ddlFile":

```
upload-model <modelName> <modelType> <ddlFile>
```

This is how you export the "Model1" DDL to the specified file "ddlFile":

```
export-ddl <ddlFile>
```

CHAPTER 5. RECORD SESSION COMMANDS IN A FILE

In this section you will learn how to use the VDB Builder CLI to record your session commands for later playback. You can use the command file to fully script the construction of your VDBs.

The recorded output is saved to a file specified in the VDB Builder's global properties configuration file (RECORDING_FILE). In the example below, the output is written to ./mySession.txt:

Figure 5.1. Code sample

```
[workspace] list
     There are no children for workspace '/workspace'.
[workspace] show-global
     Global shell properties:
     NAME                          VALUE
     ------------------------      ------------------------
     AUTO_COMMIT                   true
     EXPORT_DEFAULT_DIR            .
     IMPORT_DEFAULT_DIR            .
     RECORDING_FILE                ./myOutput.txt
     SERVER_CONNECT_ON_STARTUP     true
     SHOW_COMMAND_CATEGORY         true
     SHOW_FULL_PATH_IN_PROMPT      false
     SHOW_HIDDEN_PROPERTIES        false
     SHOW_PROP_NAME_PREFIX         false
     SHOW_TYPE_IN_PROMPT           true
[workspace] set-global RECORDING_FILE ./mySession.txt
     Successfully set global property 'RECORDING_FILE'.
[workspace] set-record on
     Recording set on at Tue Apr 26 16:18:26 CDT 2016 - File: './mySession.txt'
[workspace] create-vdb SampleVDB
     VDB 'SampleVDB' was created.
[workspace] cd SampleVDB
[SampleVDB (VDB)] add-model Model1
     Model 'Model1' was successfully added.
[SampleVDB (VDB)] cd Model1
[Model1 (Model)] add-table Table1
     Table 'Table1' was successfully added.
[Model1 (Model)] cd Table1
[Table1 (Table)] add-column Col1
     Column 'Col1' was successfully added.
[Table1 (Table)] add-column Col2
     Column 'Col2' was successfully added.
[Table1 (Table)] workspace
[workspace] set-record off
     Recording set off at Tue Apr 26 16:19:57 CDT 2016 - File: './mySession.txt'
[workspace]
```

Explanation of the Commands Used in this Example

list

shows all children at the CLI workspace context. (The workspace in this example initially contains no VDBs.)

show-global

shows the VDB Builder global constants. One of the constants specifies the recording output file (RECORDING_FILE).

set-global RECORDING_FILE ./mySession.txt

this changes the name of the file where our session commands will be saved.

set-record on

starts recording all subsequent commands.

create-vdb SampleVDB

creates the VDB called SampleVDB.

cd SampleVDB

navigates into the SampleVDB directory.

add-model Model1

adds a model called Model1.

cd Model1

navigates into Model1.

add-table Table1

adds a table called Table1

cd Table1

navigates into Table1.

add-column Col1

adds a column called Col1.

add-column Col2

adds a column called Col2.

workspace

navigates to the workspace context.

set-record off

tells the system to stop recording commands.

This is what the recording looks like:

Figure 5.2. Code sample

```
#  ----------
#  Recording set on at Tue Nov 03 15:03:29 CST 2015 - File: "./mySession.txt"
#  ----------
create-vdb SampleVDB
cd SampleVDB
add-model Model1
cd Model1
add-table Table1
cd Table1
add-column Col1
add-column Col2
workspace
#  ----------
#  Recording set off at Tue Nov 03 15:05:00 CST 2015 - File: "./mySession.txt"
#  ----------
```

CHAPTER 6. EXECUTE A COMMAND FILE

You can execute previously-saved session commands. In this example, you are going to execute the same recording that you produced above:

To play back the recording you have two options. The first of these is interactive mode. To run in interactive mode, manually remove the exit command at the end of the script, then follow these instructions:

Figure 6.1. Code sample

```
[workspace] list
     There are no children for workspace '/workspace'.
[workspace] play ./mySession.txt
     All commands in file './mySession.txt' have been executed.
[workspace] list
     Children for workspace '/workspace':

     NAME                          TYPE
     ------------------------      ------------------------
     SampleVDB                     VDB
[workspace] cd SampleVDB
[SampleVDB (VDB)] list
     Children for VDB '/workspace/SampleVDB':

     NAME                          TYPE
     ------------------------      ------------------------
     Model1                        Model
[SampleVDB (VDB)] workspace
[workspace]
```

The other option is to execute the file using the VDB Builder launch command with the -f switch. (Add an exit command at the end of the script file):

Figure 6.2. Code sample

```
[mdrilling@localhost VDBBuilder]$ ./vdbbuilder.sh -f ./mySession.txt
SLF4J implementation located in the classpath. It will be used by ModeShape for logging.
Starting engine... Started
Starting Local Repository initialization ....ModeShape version 4.4.0.Final
.ISPN000128: Infinispan version: Infinispan 'Insanely Bad Elf' 7.2.3.Final
ISPN000104: Falling back to DummyTransactionManager from Infinispan
ISPN023006: Using pure Java LevelDB implementation: org.iq80.leveldb.impl.Iq80DBFactory
.......... Started

*********************************************************************
    Welcome to VDB Builder Shell
*********************************************************************
     The following commands are supported at this context:

     cd                  commit              create-schema       create-teiid
     create-vdb          delete-schema       delete-teiid        delete-vdb
     exit                find                help                home
     import-vdb          library             list                play
     rename              rollback            set-auto-commit     set-global
     set-record          set-server          show-children       show-global
     show-status         show-summary        upload-vdb          workspace

     Enter "help" to get a list of available commands or "help <command-name>" for specific information about one command.

     To execute a specific command, try "<command-name> <args>". Also try entering a tab key for command argument completion help
.

     VDB "SampleVDB" was created.
     Model "Model1" was successfully added.
     Table "Table1" was successfully added.
     Column "Col1" was successfully added.
     Column "Col2" was successfully added.
     Shutting down...
Starting shutdown procedure ...
Shutdown completed.
     done.
[mdrilling@localhost VDBBuilder]$
```

CHAPTER 7. CONNECT TO A LOCAL RED HAT JBOSS DATA VIRTUALIZATION SERVER

With this example, you will learn how to connect to a Red Hat JBoss Data Virtualization server. A default server is defined within the VDBBuilder CLI. You can specify the server connection properties, then connect to and interact with the server. The sample session below shows how to connect to the server and view the deployed data sources.

Note that once you are connected to a server, many more server commands will become available to you. You can discover the available server commands in VDBBuilder by typing 'help' after connecting to the server.

Note

VDBBuilder will attempt to connect to the server on startup if the global property SERVER_CONNECT_ON_STARTUP is set to true.

Figure 7.1. Code sample

```
[workspace] show-status
     Current Repository Name : komodoLocalWorkspace
     Current Repository Url  : jar:file:/home/mdrilling/apps/vdbbuilder-test63/vdbbuilder/komodo-core-0.0.4.jar!/org/komodo/repository/local-repository-config.json
     Current Context : [/workspace]
     Server : [mms://localhost:9999] : [Not Connected]
[workspace] server-show-properties
     Properties for TeiidImpl '/environment/servers/DefaultServer':

     NAME                          VALUE                         DEFAULT
     ------------------------      ------------------------      ------------------------
     adminPort                     <no value>                    9999
     adminPswd                     <no value>
     adminSecure                   <no value>                    true
     adminUser                     <no value>                    admin
     connected                     <no value>
     externalLocation              <no value>
     host                          <no value>                    localhost
     jdbcPort                      <no value>                    31000
     jdbcPswd                      <no value>
     jdbcSecure                    <no value>                    false
     jdbcUser                      <no value>                    user
     lastPingTime                  <no value>
[workspace] server-connect
     Attempting to connect to 'DefaultServer' - PLEASE WAIT...
Apr 26, 2016 4:39:52 PM org.teiid.adminapi.jboss.AdminFactory createAdmin
INFO: Connected to standalone controller at localhost:9999
     Teiid 'DefaultServer' connection status: Connected
[workspace] show-status
     Current Repository Name : komodoLocalWorkspace
     Current Repository Url  : jar:file:/home/mdrilling/apps/vdbbuilder-test63/vdbbuilder/komodo-core-0.0.4.jar!/org/komodo/repository/local-repository-config.json
     Current Context : [/workspace]
     Server : [mms://localhost:9999] : [Connected]
[workspace] help
     The following commands are supported at this context:

          General Commands:
               cd                        exit                      help                      home                      library
               list                      play                      rename                    reset-global              set-auto-commit
               set-global                set-record                show-children             show-global               show-status
               show-summary              workspace

          Relational Commands:
               create-datasource         create-schema             create-vdb                delete-datasource         delete-schema
               delete-vdb                export-datasource         export-vdb                find                      import-vdb
               upload-datasource         upload-vdb

          Server Commands:
               server-datasource                 server-datasource-type            server-datasource-types           server-datasources                server-deploy-datasource
               server-deploy-driver              server-deploy-vdb                 server-disconnect                 server-get-datasource             server-get-vdb
               server-set-property               server-show-properties            server-translator                 server-translators                server-undeploy-datasource
               server-undeploy-vdb               server-unset-property             server-vdb                        server-vdbs

     Enter "help" to get a list of available commands or "help <command-name>" for specific information about one command.
     To execute a specific command, try "<command-name> <args>". Also try entering a tab key for command argument completion help.

[workspace] server-datasources
     Data sources for server 'DefaultServer':

     DashboardDS                   ExampleDS                 ModeShapeDS               OracleBQT2
     SFImporterModel               SFModel                   aVdb
[workspace] server-disconnect

     Disconnecting from 'DefaultServer'...
     Teiid 'DefaultServer' disconnected.
[workspace]
```

Explanation of the Commands Used in this Example:

show-status

display the workspace status. (Note the current server status is Not Connected.)

server-show-properties

shows the default server properties. You can use 'server-set-property' to change the properties.

server-connect

connects to the default server.

show-status

display the status again. This time, note the server status is [Connected]

help

shows the available commands. (Note the server commands that are now available.)

server-datasources

display the datasources currently deployed on the connected server.

server-disconnect

disconnects from the default server.

CHAPTER 8. INTERACT WITH VDBS ON YOUR LOCAL RED HAT JBOSS DATA VIRTUALIZATION SERVER

In this section, you will learn a few different ways in which you can interact with VDBs on your local server. You can either deploy a VDB from your VDB Builder workspace to the connected server or you can obtain a VDB from the connected server and store it in your VDB Builder workspace.

This opens up a number of possibilites for VDB interactions. For example, you can deploy a new VDB created in VDB Builder to a running server. Alternatively, you could fetch a VDB from the connected server, then edit it and re-deploy it.

Note

Deployment of dynamic VDBs from VDB Builder is somewhat limited at this time. VDB validation in VDB Builder has not yet been implemented, so you are allowed to deploy invalid VDBs. Such an attempt will fail. Also, a dynamic VDB will most likely reference other datasources. If the datasources are not present on the server, the deployment will fail. All of these issues will be addressed in the near future.

Here is how you can view the server and workspace VDBs, and deploy a workspace VDB to a running server:

Figure 8.1. Code sample

```
[workspace] show-status
     Current Repository Name : komodoLocalWorkspace
     Current Repository Url  : jar:file:/home/mdrilling/apps/vdbbuilderdv63/vdbbuilder/komodo-core-0.0.4-SN
ry-config.json
     Current Context : [/workspace]
     Server : [mms://localhost:9999] : [Connected]
[workspace] list
     Children for workspace '/workspace':

     NAME                          TYPE
     ------------------------      ------------------------
     VdbDashboardDS                VDB
[workspace] server-vdbs
     VDBs for server 'DefaultServer':

     No VDBs found on the server.
[workspace] server-deploy-vdb VdbDashboardDS
     The VDB deployed successfully.
[workspace] server-vdbs
     VDBs for server 'DefaultServer':

     VdbDashboardDS
[workspace] server-vdb VdbDashboardDS
     Details for VDB 'VdbDashboardDS' on server 'DefaultServer':

     Name: VdbDashboardDS
     Version: 1
     Type: Dynamic
     Status: ACTIVE

     VDB Models
     ------------------------
     DashboardDS

     VDB Properties:
     Name                          Value
     ------------------------      ------------------------
     deployment-name               VdbDashboardDS-vdb.xml
[workspace]
```

Explanation of the Commands Used in this Example:

show-status

display the shell status. Note the server status is Connected.

list

show all children in workspace. Note that we have one VDB VdbDashboardDS in our workspace.

server-vdbs

show all VDBs on the connected server. Note that VdbDashboardDS does not exist on the server.

server-deploy-vdb VdbDashboardDS

deploy VdbDashboardDS from VDB Builder to the connected server.

server-vdbs

again, show all VDBs on the connected server. (Note that VdbDashboardDS has been deployed.) server-vdb VdbDashboardDS: shows more detailsl for the VdbDashboardDS on the connect server.

You can also fetch a VDB from a running server and save it to the VDB Builder workspace.

Figure 8.2. Code sample

```
[workspace] show-status
     Current Repository Name : komodoLocalWorkspace
     Current Repository Url  : jar:file:/home/mdrilling/apps/vdbbuilderdv63/vdbbuilder/komodo-core-0.0.4-SNAPSHOT.jar!/or
ry-config.json
     Current Context : [/workspace]
     Server : [mms://localhost:9999] : [Connected]
[workspace] list
     There are no children for workspace '/workspace'.
[workspace] server-vdbs
     VDBs for server 'DefaultServer':

     VdbDashboardDS
[workspace] server-get-vdb VdbDashboardDS
     The VDB was copied into the workspace.
[workspace] list
     Children for workspace '/workspace':

     NAME                          TYPE
     ------------------------      ------------------------
     VdbDashboardDS                VDB
[workspace] cd VdbDashboardDS/
[VdbDashboardDS (VDB)] show-summary
     Properties for VDB '/workspace/VdbDashboardDS':

     NAME                          VALUE                                        DEFAULT
     ------------------------      ---------------------------------------      --------------------------------
     allowed-languages             <no value>
     authentication-type           <no value>
     connectionType                BY_VERSION
     deployment-name               VdbDashboardDS-vdb.xml
     description                   VdbDashboardDS, Version: 1
     gss-pattern                   <no value>
     name                          VdbDashboardDS
     originalFile                  /tko:komodo/tko:workspace/VdbDashboardDS
     password-pattern              <no value>
     preview                       false                                        false
     query-timeout                 <no value>
     security-domain               <no value>
     version                       1                                            1

     Children for VDB '/workspace/VdbDashboardDS':

     NAME                          TYPE
     ------------------------      ------------------------
     DashboardDS                   Model
[VdbDashboardDS (VDB)]
```

Explanation of the Commands Used in this Example:

show-status

display the shell status. Note the server status is Connected.

list

show all children in workspace. Note that we do not have any VDBs.

server-vdbs

show all the VDBs on the connected server. Note that VdbDashboardDS exists on the server.

server-get-vdb VdbDashboardDS

obains VdbDashboardDS from the connected server and copies it to VDB Builder workspace.

list

show all children in workspace. (Note that VdbDashboardDS is now in the VDB Builder workspace.)

cd VdbDashboardDS

navigate into the VdbDashboardDS directory.

show-summary

shows a summary (properties and children) for the current context of VdbDashboardDS.