DELPHIX

So you want to work with Delphix APIs?

September 2018

So you want to work with Delphix APIs?
You can find the most up-to-date technical documentation at:
docs.delphix.com The Delphix Web site also provides the latest product updates.
If you have comments about this documentation, submit your feedback to: infodev@delphix.com

Delphix Corp.
1400 Seaport Blvd, Suite 200
Redwood City, CA 94063

So You Want to Work with Delphix APIs?

What is RESTFul? API? JSON? CLI? Object Reference? GET/POST? Cookies? HTTP/HTTPS? cURL? Where do I begin? Who can help me? Documentation? Tutorials? That's great for Linux, but I am on Windows? Parse? Don't have a clue about sed, awk, grep, cut, and other acronyms. What about Regular Expressions, like "Hello"?

Just a small sampling of questions that you may or may not know the answers to, let alone learning Delphix and a programming language. Delphix is a technical product, and being new to the Delphix family can be a bit overwhelming. The goal for this document is simple: to enable users to get up to speed quickly on how to use Delphix APIs.

Background Information

This document assumes that you have some basic Delphix product experience and entry-level programming knowledge. The first two sections of this document, Delphix API Reference URLs and API Prerequisite Knowledge, are focused on providing the required information and reference material/URLs.

This document is for informational and demonstration purposes only. The examples are for demonstration purposes only and must be used at your own risk. As always, test and verify on development systems prior to migrating code to production environments.

What is RESTFul Programming?

http://stackoverflow.com/questions/671118/what-exactly-is-restful-programming

> A great way to learn how to generate the Delphix RESTFul API calls and the required JSON content is to use the Delphix CLI (Command Line Interface) and turn on the trace option.
>
> **`Delphix5110HWv8>`** ***setopt trace=true***
>
> All subsequent CLI commands will display the GET or POST API URL with the respective input or output JSON data string. This guide will walk you through an example later.

Delphix API Reference URLs

- CLI (Command Line Interface)
- RESTFul APIs
- Masking APIs
- Cookbook Examples

There are a number of sources available to provide details, examples, and techniques for working with Delphix APIs. This section contains a small list of URLs that are worth reviewing/reading as required.

CLI (Command Line Interface)

Delphix Documentation:

https://docs.delphix.com/display/DOCS/Command+Line+Interface+Guide

RESTFul APIs

Delphix Documentation:

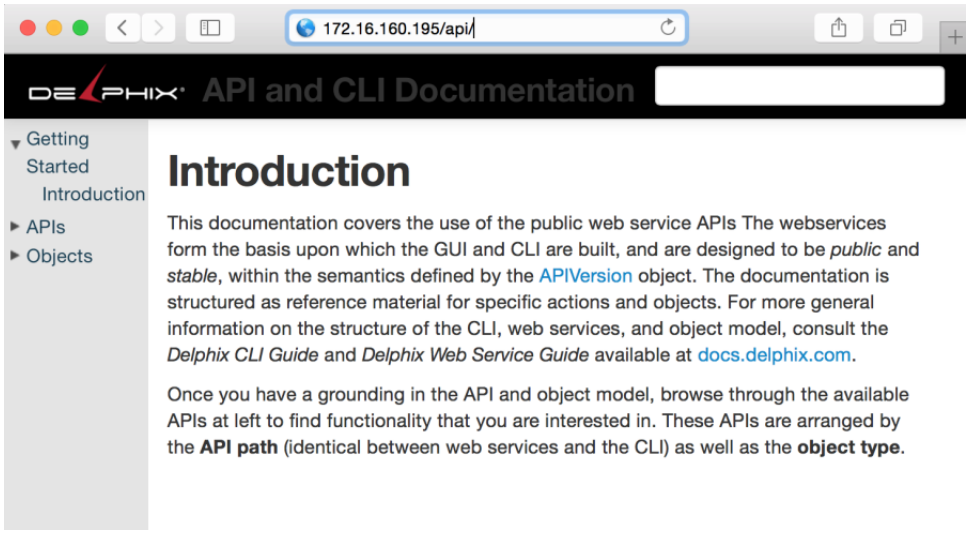
https://docs.delphix.com/display/DOCS/Web+Service+API+Guide

API Documentation is also included within the Delphix Engine using the following formula:

http://<delphix_engine>/api/

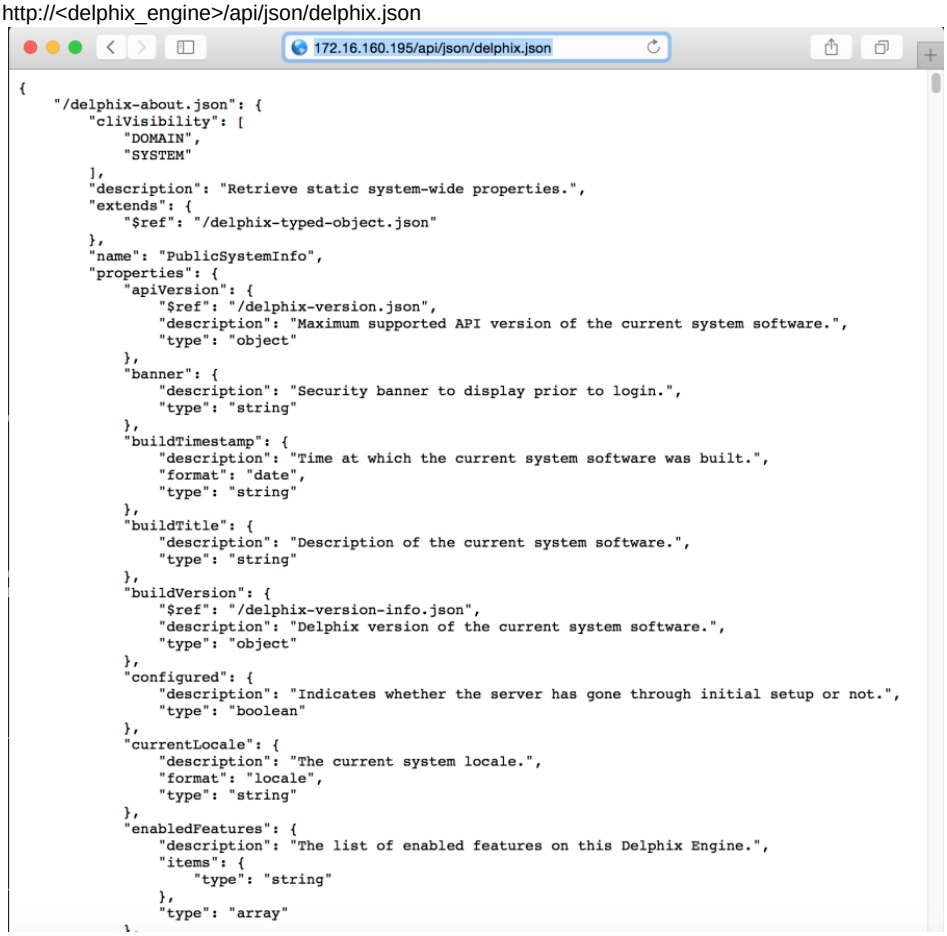
For Example: http://172.16.160.195/api/

So you want to work with Delphix APIs?



For a complete list of Delphix APIs - JSON schema format, use the following URL:

http://<delphix_engine>/api/json/delphix.json



So, looking at the first JSON key/name:

```
"/delphix-about.json": {
    "cliVisibility": [
        "DOMAIN",
        "SYSTEM"
    ],
    "description": "Retrieve static system-wide properties.",
    . . .
```

And after logging into the Delphix Engine, translating this into the URL API for about;

http://172.16.160.195/resources/json/delphix/about will respond with the returned JSON data string.

```
{"type":"OKResult","status":"OK","result":{"type":"PublicSystemInfo","prod
uctType":"standard","productName":"Delphix Engine","buildTitle":"Delphix
Engine
5.1.2.0","buildTimestamp":"2016-09-02T22:28:43.000Z","buildVersion":{"type
":"VersionInfo","major":5,"minor":1,"micro":2,"patch":0},"configured":true
,"enabledFeatures":["XPP","JETSTREAM"],"apiVersion":{"type":"APIVersion","
major":1,"minor":8,"micro":0},"banner":null,"locales":["en-US"],"currentLo
cale":"en-US"},"job":null,"action":null}
For now, just remember that the Delphix Engine contains the API
Documentation and Delphix JSON schema.
```

Masking APIs

Please note:

> The Masking APIs are currently being overhauled to provide a complete and robust set of APIs. If you do not see the APIs you need in order to perform a certain function, contact Delphix support personnel; there may already be some internal solution that has not yet been certified for general acceptance releases.

https://docs.delphix.com/display/DOCS/Masking+API+Calls+to+Run+a+Masking+Job

Cookbook Examples

Delphix documentation also includes a number of cookbook examples that will not be duplicated in this document, but may be referenced.

https://docs.delphix.com/display/DOCS/API+Cookbook%3A+Common+Tasks%2C+Workflows%2C+and+Examples

There are also working examples provided within this document and available for download as well. See Appendix for a complete list.

API Prerequisite Knowledge

- JSON
  - Keys and Values
  - Types of Values
  - Numbers, Booleans, and Strings.
  - Null values
  - Arrays
  - Objects
  - Summary

JSON

JSON (JavaScript Object Notation) is a minimal, readable format for structuring data. It is a simple format for transmitting data between applications, as an alternative to XML. The Delphix API uses JSON data structure in the format of strings to send and receive data from the API calls, as you will see later in the examples. First, let's look at the JSON fundamentals.

Keys and Values

The two primary parts that make up JSON are keys and values. Together they make key/value pairs, also called name/value pairs.

- **Key** – Always a string enclosed in quotation marks.
- **Value** – Can be a string, number, boolean expression, array, or object.
- **Key/Value Pair** – Follows a specific syntax, with the key followed by a colon followed by the value. Key/value pairs are comma separated.

Let's take a JSON sample string and identify each part of the code.

```
{
    "foo" : "bar",
    "rows" : 100
}
```

The curly brackets start and end the string. The key is "foo" and the value is "bar". A colon (:) is the delimiter between them. A comma (,) is the delimiter for multiple key/value pairs. The second pair is "rows" and the value is a number of 100.

Types of Values

Number	An integer or decimal number
Boolean	True or false
String	plain text alphanumeric readable characters
Null	Empty
Array	An associative array of values
Object	An associative array of key/value pairs

Numbers, Booleans, and Strings.

It is very important to understand the APIs JSON object definitions. Quoted values are treated as strings!

"x" : "1" is treated as a string, while

"x" : 1 is treated as a number

"y" : "true" is treated as a string, while

"y" : true is treated as a boolean true (false)

Null values

```
{
    "z" :
  , "b" : "World"
}
```

Nulls are empty values, but sometimes programmers code "" as a null value.

```
{
    "z" : ""
  , "b" : "World"
  }
```

So always verify how the null values are defined and handled by the application.

Arrays

An array is indicated with the square brackets: [value1, value2, etc.]. In this example, we have added a categories key with an array of values.

```
...
"foo" : {
  "bar" : "Hello",
  "category" : [ "greetings", "morals" ]
}
...
```

Objects

An object is indicated by curly brackets: {"key", "value"}. Everything inside of the curly brackets is part of the object. We already learned that a value could be an object. Therefore, "foo" and the corresponding object are a key/value pair.

```
...
"foo" : { "bar" : "Hello" }
...
```

The key/value pair "bar" : "Hello" is nested inside the key/value pair "foo" : { ... }. That is an example of a hierarchy (or nested data) within JSON data.

Arrays and Objects can be nested or contained within the same level.

Summary

JSON arrays are `[ , , ]`

JSON nested objects are `, , "x":{ "a":"1", "b":"2" }, ,`

JSON data can be passed within the HTTP URL (file or argument), the header, or other handlers.

From within Shell Scripts or Programming Languages, JSON data is typically processed though a "JSON parser." This topic is covered later.

Delphix CLI

Connecting to the Delphix Engine CLI

Reference: https://docs.delphix.com/display/DOCS/Connecting+to+the+CLI

There are two user roles accessible, the **sysadmin** and the **delphix_admin**.

From a shell environment, you can connect using the ssh command. The IP Address (or Hostname) represents the Delphix Engine (case sensitive):

```
ssh sysadmin127.16.160.195
```

```
ssh delphix_admin127.16.160.195
```

From a putty session, open an ssh connection to the Delphix Engine IP Address or Hostname (case sensitive):

```
open 127.16.160.195
```

```
Login User: sysadmin@SYSTEM
```

`#...` or `...`

```
Login User: delphix_admin@DOMAIN
```

After entering the correct password for the respective user, the menus for that user's role will be different. For example, the **sysadmin@SYSTEM** user has engine storage, whereas the **delphix_admin@DOMAIN** user has database provisioning.

You can use the CLI for scripting and configure the connection for ssh passwordless connections.

https://docs.delphix.com/display/DOCS/CLI+Cookbook%3A+Configuring+Key-Based+SSH+Authentication+for+Automation

How to use the CLI to learn the APIs

As stated earlier, a great way to learn how to generate the Delphix RESTFul API calls and the required JSON content is to use the Delphix CLI (Command Line Interface) and turn on the `CLI> setopt trace=true` option.

Below is an example of how to get the JSON-required parameters for a database refresh per the type of refresh performed.

> Other types or options may require other JSON parameters, so after changing any parameter, we recommend performing an "ls" command to see if there are any new parameters and/or required values.

The refresh database example below shows how to use the CLI to identify reference objects for other CLI commands and the respective RESTFul API structure when the ***setopt trace=true*** option is set.

```
$ ssh delphix_admin172.16.160.195
Password:
Delphix5030HWv8> ls
Children
about
action
...
connectivity
database
environment
...
toolkit
user

Operations
version
Delphix5030HWv8> database
Delphix5030HWv8 database> ls
Objects
NAME   PROVISIONCONTAINER DESCRIPTION
DPXDEV01  -
Vdelphix_demo delphix_demo  -
delphix_demo -
Scripts   -
V_2C1   Scripts    -
Vvfiles   -     -

Children
template

Operations
createEmpty
createRestorationDataset
export
fileMapping
link
oracleSupportedCharacterSets
provision
validateXpp
xpp
```

First, we need to identify the target Delphix virtualized database object to refresh ...

> Each Delphix object has a reference that is typically used for parameter values.

```
Delphix5030HWv8 database> select Vdelphix_demo
Delphix5030HWv8 database 'Vdelphix_demo'> ls
Properties
```

```
 type: MSSqlDatabaseContainer
 name: Vdelphix_demo
 creationTime: 2016-06-16T14:30:03.033Z
 currentTimeflow: 'DB_PROVISION@2016-06-16T10:30:08'
 delphixManaged: true
 description: (unset)
 group: Windows
 masked: false
 os: Windows
 performanceMode: DISABLED
 processor: x86
 provisionContainer: delphix_demo
 reference: MSSQL_DB_CONTAINER-39
 restoration: false
 runtime:
  type: MSSqlDBContainerRuntime
  logSyncActive: false
 sourcingPolicy:
  type: SourcingPolicy
  loadFromBackup: false
  logsyncEnabled: false
 transformation: false

Operations
delete
...
purgeLogs
refresh
removeLiveSource
...
Delphix5030HWv8 database 'Vdelphix_demo'> refresh
Delphix5030HWv8 database 'Vdelphix_demo' refresh *> ls
Properties
 type: RefreshParameters
 timeflowPointParameters:
  type: TimeflowPointSemantic
  container: (required)
  location: LATEST_POINT
Delphix5030HWv8 database 'Vdelphix_demo' refresh *> set
timeflowPointParameters.container=delphix_demo
Delphix5030HWv8 database 'Vdelphix_demo' refresh *> ls
Properties
 type: RefreshParameters
 timeflowPointParameters:
  type: TimeflowPointSemantic
  container: delphix_demo (*)
  location: LATEST_POINT
Delphix5030HWv8 database 'Vdelphix_demo' refresh > *commit
 Dispatched job JOB-100
 DB_REFRESH job started for "Windows/Vdelphix_demo".
 Validating that this dataset is managed by Delphix.
 Stopping virtual database.
 Unmounting datasets.
```

```
Unexporting storage containers.
Metadata for dSource "Vdelphix_demo" successfully deleted.
Starting provisioning of virtual database "Vdelphix_demo".
Creating new TimeFlow.
Generating recovery scripts.
Mounting datasets.
Mounting read-only source logs dataset.
Running user-specified pre-provisioning script.
Recovering virtual database.
The virtual database recovery was successful.
Unmounting read-only source logs dataset.
```

```
 Running user-specified post-provisioning script.
 The virtual database "Vdelphix_demo" was successfully provisioned.
 DB_REFRESH job for "Windows/Vdelphix_demo" completed successfully.
```

Refresh again but this time turn on the **`setopt trace=true`** option.

```
Delphix5030HWv8 database 'Vdelphix_demo'> refresh
Delphix5030HWv8 database 'Vdelphix_demo' refresh *> ls
Properties
 type: RefreshParameters
 timeflowPointParameters:
  type: TimeflowPointSemantic
  container: (required)
  location: LATEST_POINT
Delphix5030HWv8 database 'Vdelphix_demo' refresh *> set
timeflowPointParameters.container=delphix_demo
Delphix5030HWv8 database 'Vdelphix_demo' refresh *> ls
Properties
 type: RefreshParameters
 timeflowPointParameters:
  type: TimeflowPointSemantic
  container: delphix_demo
  location: LATEST_POINT
Delphix5030HWv8 database 'Vdelphix_demo' refresh *> setopt trace=true
Delphix5030HWv8 database 'Vdelphix_demo' refresh *> commit
=== POST /resources/json/delphix/database/MSSQL_DB_CONTAINER-39/refresh ===
{
 "type": "RefreshParameters",
 "timeflowPointParameters": {
  "type": "TimeflowPointSemantic",
  "container": "MSSQL_DB_CONTAINER-38"
 }
}
...
```

> The "container" value in the JSON output above is different from the target VDB reference, because we are refreshing from the source database container! In this example, the **set timeflowPointParameters.container=delphix_demo** is represented in JSON output as **"container": "MSSQL_DB_CONTAINER-38"**

Using the CLI, you can identify the RESTFul API POST and GET commands along with the JSON input data requirements.

```
=== POST /resources/json/delphix/database/MSSQL_DB_CONTAINER-39/refresh ===
{
 "type": "RefreshParameters",
 "timeflowPointParameters": {
  "type": "TimeflowPointSemantic",
  "container": "MSSQL_DB_CONTAINER-38"
 }
}
```

So framing the RESTFul URL for a virtual database refresh, the URL will look like

http://<delphix_engine>/resources/json/delphix/database/ MSSQL_DB_CONTAINER-39 /refresh

where the **MSSQL_DB_CONTAINER-39** represents the target virtualized database to refresh. We need to POST the JSON data to the URL for processing.

```
{
 "type": "RefreshParameters",
 "timeflowPointParameters": {
  "type": "TimeflowPointSemantic",
  "container": "MSSQL_DB_CONTAINER-38"
 }
}
```

The **"timeflowPointParameters"** key has 6 **"type": "..."** options, each of which has its own set of parameters. The type **"TimeflowPointSemantic"** uses the default LATEST_POINT within the source container, so for simplicity we will use this type. For more information on timeflowPointParameters 6 types, see the Advanced Section.

If this is a little confusing at this point, do not worry, that's typical. Complete examples will be shown later. The important items to remember are:

- Delphix often uses object reference names within the JSON data.
- Using the **`setopt trace=true`** option provides the construct for the RESTFul API URLs and the JSON data for POST / GET operations.

HTTP

We use the HTTP protocol every day for web browsing and commercial business. From finding a new restaurant to buying a 1986 Ford Thunderbird Turbo Coupe!

Most people see the HTTP within the URL Address field within the Web Browser window – for example, http://www.google.com

But behind the scenes, HTTP is performing a wide range of functionality. For RESTFul APIs, they use HTTP's GET and POST form functionality to process data. In Delphix's case, the data is also represented as JSON structures.

HTTP GET operation is used to return data only, while HTTP POST operations is used to provide data input in the form of a structured JSON data string or file.

cURL

What is cURL?

https://en.wikipedia.org/wiki/CURL

The cURL client command is based on a library supporting a number of web protocols, including HTTP. The "curl" command can be called from the command line, while the cURL library is commonly integrated with your favorite programming languages, such as Java, JSP, Python, Perl, PHP, .NET, and PowerShell.

Due to its widespread adoption, we will use cURL for making the Delphix RESTFul API calls within this document. Some operating systems or languages support their own HTTP commands / related libraries, and you can use these instead of cURL. One alternative is the "wget" command described later.

Is cURL installed?

```
Operating System Prompt> curl --version
curl 7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.19.1 Basic ECC
zlib/1.2.3 libidn/1.18 libssh2/1.4.2
Protocols: tftp ftp telnet dict ldap ldaps http file https ftps scp sftp
Features: GSS-Negotiate IDN IPv6 Largefile NTLM SSL libz
```

Get the HTTP output from google.com

```
Operating System Prompt> curl www.google.com
```

Wget

An alternative to cURL is Wget, which is typically a native command on all Linux environments. See the Appendix for a complete comparison between Wget and cURL.

dxtoolkit2

Delphix has developed a very robust toolkit, dxtoolkit2, which utilizes the Delphix RESTFul APIs. This toolkit is cross-platform. Its commands are built with the Perl programming language.

We recommend that you review the dxtoolkit2 documentation; you may find a utility that already performs your desired function. For example, the utility ***dx_get_analytics*** is absolutely great for dumping analytic data from the Delphix Engine into a .csv (comma separated value) format, which you can then easily integrate into your enterprise monitoring tools. See the sample "Analytics" use case.

Contact Delphix personnel for the latest download.

For the dxtoolkit2 README file Table of Contents, see the Appendix.

Delphix RESTFul APIs Command Line Basics

Authentication

RESTFul APIs require authentication. Just plugging the URL into a web browser or running an operating system cURL command will return an authentication/login required error message.

Command Line:

curl http://172.16.160.195/resources/json/delphix/environment

Response:

```
<!DOCTYPE html><html><head><title>Apache Tomcat/8.0.29 - Error
report</title><style type="text/css">H1
{font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;
font-size:22px;} H2
{font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;
font-size:16px;} H3
{font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;
font-size:14px;} BODY
{font-family:Tahoma,Arial,sans-serif;color:black;background-color:white;} B
{font-family:Tahoma,Arial,sans-serif;color:white;background-color:#525D76;}
P
{font-family:Tahoma,Arial,sans-serif;background:white;color:black;font-siz
e:12px;}A {color : black;}A.name {color : black;}.line {height: 1px;
background-color: #525D76; border: none;}</style> </head><body><h1>HTTP
Status 403 - Use /resources/json/delphix/login to log in first</h1><div
class="line"></div><p><b>type</b> Status report</p><p><b>message</b> <u>Use
/resources/json/delphix/login to log in first</u></p><p><b>description</b>
<u>Access to the specified resource has been forbidden.</u></p><hr
class="line"><h3>Apache Tomcat/8.0.29</h3></body></html>
```

The authentication process requires you to establish a session first.

The example within this section illustrates the session/login and subsequent API calls with cURL using cookies created when the session was established.

Session

API Version Information

When programming for compatibility, the API version number is very important. Please be aware of differences between versions for enterprise applications. For example, if you specify a 1.7.0 version, ONLY the available calls and functionality for that version will be used, and it will only be operational on Delphix Engine versions that support that version.

Only the session uses the –c for the cookie creation. The other commands use the –b for using the existing cookie!!!

From the Unix/Linux command line

```
curl -s -X POST -k --data @-
http://172.16.160.195/resources/json/delphix/session \
 -c ~/cookies.txt -H "Content-Type: application/json" <<EOF
{
 "type": "APISession",
 "version": {
  "type": "APIVersion",
  "major": 1,
  "minor": 7,
  "micro": 0
 }
}
EOF
```

Returned to the command line are the results (added linefeeds for readability)

```
{
"type":"OKResult",
"status":"OK",
"result":{
 "type":"APISession",
 "version":{
  "type":"APIVersion",
  "major":1,
  "minor":7,
  "micro":0
 },
 "locale":null
 ,"client":null
 }
,"job":null
,"action":null
}
```

Login

Once you have established the session, the next step is to authenticate to the server by executing the Login Request API. Unauthenticated sessions are prohibited from making any API calls other than this login request. The username can be either a system user or domain user; the backend will authenticate using the appropriate method.

> **Only the session uses the –c for the cookie creation. The login and other commands use the –b for using the existing cookie!!!**

```
curl -s -X POST -k --data @-
http://172.16.160.195/resources/json/delphix/login \
 -b ~/cookies.txt -H "Content-Type: application/json" <<EOF
{
 "type": "LoginRequest",
 "username": "delphix_admin",
 "password": "delphix"
}
EOF
```

Returned to the command line are the results (added linefeeds for readability)

```
{
"status":"OK",
"result":"USER-2",
"job":null,
"action":null
}
```

Sample Delphix API call

With a successful authentication (session, login and saved cookie), calls to the Delphix Engine can now be made to perform the desired functionality.

> **Only the session uses the –c for the cookie creation. The login and other commands use the –b for using the existing cookie!!!**

For starters, let's create a session, login, and get the existing environments defined within the Delphix Engine.

```
curl -s -X POST -k --data @-
http://172.16.160.195/resources/json/delphix/session \
-c ~/cookies.txt -H "Content-Type: application/json" <<EOF
{
 "type": "APISession",
 "version": {
  "type": "APIVersion",
  "major": 1,
  "minor": 7,
  "micro": 0
 }
}
EOF
```

```
curl -s -X POST -k --data @-
http://172.16.160.195/resources/json/delphix/login \
 -b ~/cookies.txt -H "Content-Type: application/json" <<EOF
{
 "type": "LoginRequest",
 "username": "delphix_admin",
 "password": "delphix"
}
EOF
```

```
curl -X GET -k http://172.16.160.195/resources/json/delphix/environment \
 -b ~/cookies.txt -H "Content-Type: application/json"
```

Returned to the command line are the results (added linefeeds for readability)


```
{
"type":"ListResult",
"status":"OK",
"result":
[
 {"type":"WindowsHostEnvironment",
  "reference":"WINDOWS_HOST_ENVIRONMENT1",
  "namespace":null,
  "name":"Window Target",
  "description":"",
  "primaryUser":"HOST_USER-1",
  "enabled":false,
  "host":"WINDOWS_HOST1",
  "proxy":null
 },
 {
  "type":"UnixHostEnvironment",
  "reference":"UNIX_HOST_ENVIRONMENT-3",
  "namespace":null,
  "name":"Oracle Target",
  "description":"",
  "primaryUser":"HOST_USER-3",
  "enabled":true,
  "host":"UNIX_HOST-3","aseHostEnvironmentParameters":null
 }
],
"job":null,
"action":null,
"total":2,
"overflow":false
}
```

Windows PowerShell Authentication Example

See the PowerShell section below if cURL is not yet available on your operating system.

These commands work on Windows Command Prompt with the respective JSON files:

- session.json and login.json
- Filename: session.json

```
{
 "type": "APISession",
 "version": {
  "type": "APIVersion",
  "major": 1,
  "minor": 7,
  "micro": 0
 }
}
```

Filename: login.json

```
{
"type": "LoginRequest",
"username": "delphix_admin",
"password": "delphix"
}
```

(Powershell you use curl.exe or modify the default alias)...

```
curl.exe --insecure -c cookies.txt -sX POST -H "Content-Type:
application/json" -d "@session.json"
http://172.16.160.195/resources/json/delphix/session
curl.exe --insecure -b cookies.txt -sX POST -H "Content-Type:
application/json" -d "@login.json"
http://172.16.160.195/resources/json/delphix/login
curl.exe --insecure -b cookies.txt -sX GET -H "Content-Type:
application/json" -k http://172.16.160.195/resources/json/delphix/system
```

Putting the above commands within a Powershell script:
Filename: auth1.ps1

```
Filename: auth.ps1
Description: Delphix Powershell Sample Authentication Script ...
Date: 2016-08-02
Author: Bitt...
```

Variables ...

```
$nl = [Environment]::NewLine
$BaseURL = "http://172.16.160.195/resources/json/delphix"
$cookie = "cookies.txt"
```

Session JSON Data ...

```
write-output "${nl}Creating session.json file ..."
$json = @"
{
 "type": "APISession",
 "version": {
  "type": "APIVersion",
  "major": 1,
  "minor": 7,
  "micro": 0
 }
}
"@
```

Output File using UTF8 encoding ...

```
write-output $json | Out-File "session.json" -encoding utf8
```

Delphix Curl Session API ...

```
write-output "${nl}Calling Session API ...${nl}"
$results = (curl --insecure -c "${cookie}" -sX POST -H "Content-Type:
application/json" -d "@session.json" -k ${BaseURL}/session)
write-output "Session API Results: ${results}"
```

Login JSON Data ...

```
write-output "${nl}Creating login.json file ..."
$user = "delphix_admin"
$pass = "delphix"
$json = @"
{
 "type": "LoginRequest",
 "username": "${user}",
 "password": "${pass}"
}
"@
```

Output File using UTF8 encoding ...

```
write-output $json | Out-File "login.json" -encoding utf8
```

Delphix Curl Login API ...

```
write-output "${nl}Calling Login API ...${nl}"
$results = (curl --insecure -b "${cookie}" -sX POST -H "Content-Type:
application/json" -d "@login.json" -k ${BaseURL}/login)
write-output "Login API Results: ${results}"
```

Delphix Curl system API ...

```
write-output "${nl}Calling System API ...${nl}"
$results = (curl --insecure -b "${cookie}" -sX GET -H "Content-Type:
application/json" -k ${BaseURL}/system)
write-output "System API Results: ${results}"
```

The end is near ...

```
echo "${nl}Done ...${nl}"
exit;
```

Sample Powershell Script Output:

```
PS> . .\auth.ps1
Creating session.json file ...
Calling Session API ...
Session API Results:
{"type":"OKResult","status":"OK","result":{"type":"APISession","version":{
"type":"APIVersion","major":1,"minor":7,"micro":0},"locale":null,"client":
null},"job":null,"action":null}
Creating login.json file ...
Calling Login API ...
Login API Results:
{"type":"OKResult","status":"OK","result":"USER2","job":null,"action":null}
Calling System API ...
System API Results:
{"type":"OKResult","status":"OK","result":{"type":"SystemInfo","productTyp
e":"standard","productName":"Delphix Engine","buildTitle":"Delphix Engine
5.1.1.0","buildTimestamp":"20160721T07:23:41.000Z","buildVersion":{"type":
"VersionInfo","major":5,"minor":1,"micro":1,"patch":0},"configured":true,"
enabedFeatures":["XPP","MSSQLHOOKS"],"apiVersion":{"type":"APIVersion","ma
jor":1,"minor":8,"micro":0},"banner":null,"locals":["enUS"],"currentLocale
":"enUS","hostname":"Delphix5110HWv8","sshPublicKey":"ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDOsrp7Aj6hFQh9yBq7273B+qtPKmCu1B18nPvr08yjt/
IZeM4qKk7caxExQS9rpfU8AWoT7e8ESV7NkBmUzOHrHnLsuJtxPqeYoqeMubVxYjJuxlH368sZ
uYsnB04KM0mi39e15lxVGvxQk9tyMpl7gs7cXRz1k6puncyiczU/axGq7ALHU2uyQoVmlPasuH
Jbq23d21VAYLuscbtgpZLAFlR8eQH5Xqaa0RT+aQJ6B1ihZ7S0ZN914M2gZHHNYcSGDWZHwUnB
Gttnxx1ofRcyN4/qwT5iHq5kjApjSaNgSAU0ExqDHiqgTq0wttf5nltCqGMTFR7XY38HiNq++a
tDroot@Delphix5110HWv8\n","memorySize":8.58107904E9,"platform":"VMware with
BIOS date
05/20/2014","uuid":"564d7e1df4cb-f91098fd348d74817683","processors":[{"typ
e":"CPUInfo","speed":2.5E9,"cores":1}],"storageUsed":2.158171648E9,"storag
eTotal":2.0673724416E10,"installationTime":"2016-07-27T13:28:46.000Z"},"jo
b":null,"action":null}
Done ...
PS>
```

API Shell Scripts Programming Language Examples

Why Use Shell Scripts?

Shell scripts are great tools for rapid development and validation for simple (smaller) requirements. Most development is done iteratively, and shell scripts provide immediate feedback on logic and code.

> **Company-supported programming languages are the preferred tools for enterprise applications.**

Most likely, your company employs more Java and/or PHP programmers than Linux Shell script programmers.

The "use cases" will be programmed using either Unix/Linux Shell and/or Windows PowerShell scripts. You can easily port the logic from these scripts into your favorite programming language. Basic examples of connection with API will be provided for a number of major programming languages in a later section of this document.

Linux/Unix/(and Mac too) Shell Scripts

There are numerous shell environments, including sh, bash, csh, ksh, and tsh. Identify the current shell environment using any of the commands below:

```
ps -p $$
ps -p $$ -ocomm=
echo $0
ps -ef | grep $$ | grep -v grep
ps -ef | egrep "^\s*\d+\s+$$\s+"
```

The examples provided have all been run from the bash shell environment and may or may not run the same as the other shells. We recommend that you start a bash shell by typing **bash** at the operating system prompt, like this:

```
Operating_System_Prompt> bash
```

The scripts included within this document have all been run on Linux and Mac environments within a bash shell and are NOT certified by any means. As always, test and verify in development for your environment.

For non-Linux platforms and non-bash shell environments, please re-validate for your configuration and search the web for any alternative methods/tools/utilities that may perform the same actions.

Windows PowerShell

> Powershell Open Source is now available for Linux and Mac OS.
>
> https://techcrunch.com/2016/08/18/microsoft-open-sources-powershell-brings-it-to-linux-and-os-x/

Requirements

Windows has a number of versions of Powershell. The minimum version for Delphix is 2.0 for SQL Server 2008 environments. There are numerous enhancements and features with subsequent Powershell versions. Additionally, you must be aware of the architecture of 32bit or 64bit Powershell versions you are running from within.

```
PS> $PSVersionTable.PSVersion

Major Minor Build Revision
----- ----- ----- --------
2  0  -1  -1
```

32bit or 64 bit

If executing Powershell scripts from within Delphix Pre/Post Scripts commands or Delphix hooks, the default Powershell used is 32 bit, whereas the typical default Windows Powershell is 64 bit. However, Powershell allows you to execute 64 bit Powershell command from within the 32 bit environment. Shown below is a simple alias, ps64, to execute 64bit Powershell scripts. \

```
PS> set-alias ps64
"$env:windir\sysnative\WindowsPowerShell\v1.0\powershell.exe"
```

Sample call to execute 64bit Powershell script

```
PS> ps64 [path\to\any_64bit_powershell_script].ps1
```

Courtesy of this article: http://www.gregorystrike.com/2011/01/27/how-to-tell-if-powershell-is-32-bit-or-64-bit/

```
PS> if ($env:Processor_Architecture -eq "x86") { write "running on 32bit" }
else {write "running on 64bit"}
running on 32bit
```

. . . or . . .

```
PS> if ([System.IntPtr]::Size -eq 4) { "32-bit" } else { "64-bit" }
32-bit
```

It is worth noting that the locations of the 32-bit and 64-bit versions of Powershell are somewhat misleading. The 32-bit PowerShell is found at C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe

and the 64-bit PowerShell is at C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

Execution of Scripts Security Disabled

It is possible to disable Powershell environments on the system. If they are disabled, you will see the following error for any Powershell script that you try to execute.

```
PS> . .
[any_powershell_script].ps1
File [any_powershell_script].ps1 cannot be loaded because the execution of
scripts is disabled on this system. Please see "get-help about signing" for
more details.
At line:1 char:2
+ . <<<< .\ [any_powershell_script].ps1
 + CategoryInfo   : NotSpecified: ( [], PSSecurityException
 + FullyQualifiedErrorId : RuntimeException
```

To enable Powershell scripts to be executed, set the execution policy to Yes.

```
PS> set-executionpolicy remotesigned
Execution Policy Change
The execution policy helps protect you from scripts that you do not trust.
Changing the execution policy might expose you to the security risks
described in the about_Execution_Policies help topic. Do you want to change
the execution policy?
[Y] Yes  [N] No  [S] Suspend  [?] Help (default is "Y"): Y
PS>
```

Now your shell scripts will be executed.

curl.exe

> Not all Windows platforms have the cURL executable installed. The easiest method I found was to install the git+ client for Windows.
> https://git-for-windows.github.io/

The Git install includes, among other things, curl.exe. After installing, the /mingw64/bin will be added to your PATH. Then you will be able to use the curl command from the Windows Command Prompt or PowerShell console.

```
PS> which curl.exe
/mingw64/bin/curl

PS> curl.exe --version
curl 7.49.1 (x86_64-w64-mingw32) libcurl/7.49.1 OpenSSL/1.0.2h zlib/1.2.8
libidn/1.32 libssh2/1.7.0 nghttp2/1.12.0 librtmp/2.3
Protocols: dict file ftp ftps gopher http https imap imaps ldap ldaps pop3
pop3s rtmp rtsp scp sftp smtp smtps telnet tftp
Features: IDN IPv6 Largefile SSPI Kerberos SPNEGO NTLM SSL libz TLS-SRP
HTTP2 Metalink
```

Invoking the curl or curl.exe from Powershell command line.

```
PS> Get-Command curl

CommandType  Name      ModuleName
-----------  ----      ----------
Alias   curl -> Invoke-WebRequest


PS> Get-Command curl.exe


CommandType  Name      ModuleName
-----------  ----      ----------
Application  curl.exe
```

If the alias curl name is to the Invoke-WebRequest, you will need to use the curl.exe command explicitly or remove the alias.

```
PS> Remove-item alias:curl
```

Verify that curl and/or curl.exe work from the respective Powershell environment:

```
PS> curl.exe --version
curl 7.49.1 (x86_64-w64-mingw32) ...

PS> curl --version
curl 7.49.1 (x86_64-w64-mingw32) ...
```

JSON Parsing

Unix/Linux/Mac Shell

http://stackoverflow.com/questions/1955505/parsing-json-with-unix-tools

https://gist.github.com/cjus/1047794

Unix/Linux tools come natively with a host of shell utilities that one can use for parsing out the desired name/value pairs. Tools include sed, awk, cut, tr, and grep, to name a few. System administrators use these utilities frequently and may be able to assist with the methods for parsing JSON strings.

Basic awk and sed parsing

```
json='{"type":"OKResult","status":"OK","result":{"type":"Job","reference":
"JOB-53","namespace":null,"name":null,"actionType":"DB_SYNC","target":"ORA
CLE_DB_CONTAINER-9","targetObjectType":"OracleDatabaseContainer","jobState
":"RUNNING","startTime":"2016-08-12T19:58:59.811Z","updateTime":"2016-08-1
2T19:58:59.828Z","suspendable":true,"cancelable":true,"queued":false,"user
":"USER-2","emailAddresses":null,"title":"Run SnapSync for database
\"VDPXDEV1\".","percentComplete":0.0,"targetName":"Oracle_Source/VDPXDEV1"
,"events":[{"type":"JobEvent","timestamp":"2016-08-12T19:58:59.840Z","stat
e":null,"percentComplete":0.0,"messageCode":"event.job.started","messageDe
tails":"DB_SYNC job started for
\"Oracle_Source/VDPXDEV1\".","messageAction":null,"messageCommandOutput":n
ull,"diagnoses":[],"eventType":"INFO"}],"parentActionState":"WAITING","par
entAction":"ACTION-238"},"job":null,"action":null}'

echo $json | sed -e 's/[{}]/''/g' | awk -v RS=',' -F: '{print $1 $2}'
"type""OKResult"
"status""OK"
"result""type"
"reference""JOB-53"
"namespace"null
"name"null
"actionType""DB_SYNC"
"target""ORACLE_DB_CONTAINER-9"
"targetObjectType""OracleDatabaseContainer"
"jobState""RUNNING"
"startTime""2016-08-12T19
"updateTime""2016-08-12T19
"suspendable"true
"cancelable"true
"queued"false
"user""USER-2"
"emailAddresses"null
"title""Run SnapSync for database \"VDPXDEV1\"."
"percentComplete"0.0
"targetName""Oracle_Source/VDPXDEV1"
"events"["type"
"timestamp""2016-08-12T19
"state"null
"percentComplete"0.0
"messageCode""event.job.started"
"messageDetails""DB_SYNC job started for \"Oracle_Source/VDPXDEV1\"."
"messageAction"null
"messageCommandOutput"null
"diagnoses"[]
"eventType""INFO"]
"parentActionState""WAITING"
"parentAction""ACTION-238"
"job"null
"action"null
```

Find jobState. Print the second argument, and remove the double quotes.

```
echo $json | sed -e 's/[{}]/''/g' | sed s/\"//g | awk -v RS=',' -F:
'$1=="jobState"{print $2}'
RUNNING
```

The first sed removed the brackets and braces. The second sed removes the double quotes. The awk command parses the line by comma delimiters and then parses the each line by the semi-colon delimiter and if the first variable $1 is equal to the **jobState** value then print the second $2 variable.

If the results contain an array of values, then you need to loop through each set and parse out the desired value. For example,

```
json='
{"type":"ListResult","status":"OK","result":[{"type":"WindowsHostEnvironme
nt","reference":"WINDOWS_HOST_ENVIRONMENT-1","namespace":null,"name":"Wind
ow
Target","description":"","primaryUser":"HOST_USER-1","enabled":false,"host
":"WINDOWS_HOST-1","proxy":null},{"type":"UnixHostEnvironment","reference"
:"UNIX_HOST_ENVIRONMENT-3","namespace":null,"name":"Oracle
Target","description":"","primaryUser":"HOST_USER-3","enabled":true,"host"
:"UNIX_HOST-3","aseHostEnvironmentParameters":null}],"job":null,"action":n
ull,"total":2,"overflow":false}'
```

Parse out array object into separate lines

```
SOURCE_ENV="Oracle Target"
lines=`echo ${json} | cut -d "[" -f2 | cut -d "]" -f1 | awk -v RS='},{}'
-F: '{print $0}' `
while read -r line
do
 #echo "Processing $line"
 #echo $line | sed -e 's/[{}]/''/g' | sed s/\"//g | awk -v RS=',' -F:
'$1=="name"{print $2}'
 TMPNAME=`echo $line | sed -e 's/[{}]/''/g' | sed s/\"//g | awk -v RS=','
-F: '$1=="name"{print $2}' `
 #echo "Name: |${TMPNAME}| |${SOURCE_ENV}|"
 if [[ "${TMPNAME}" == "${SOURCE_ENV}" ]]
 then
  echo $line | sed -e 's/[{}]/''/g' | sed s/\"//g | awk -v RS=',' -F:
'$1=="primaryUser"{print $2}'
  PRI_USER=`echo $line | sed -e 's/[{}]/''/g' | sed s/\"//g | awk -v RS=','
-F: '$1=="primaryUser"{print $2}' `
  break
 fi
done <<< "$(echo -e "$lines")"


echo "primaryUser reference: ${PRI_USER}"
```

Output:

```
primaryUser reference: HOST_USER-3
```

The above methods will be used within the sample scripts since they use the native Linux tools. They typically do not require you to load extra packages or libraries onto the system.

There are a number of open source utilities designed to simplify the parsing of JSON, such as jsawk and jq.

jsawk

Linux:

- https://github.com/micha/jsawk

Mac:

- http://brewformulas.org/jsawk
- http://macappstore.org/jsawk/
- http://johnattebury.com/blog/2011/06/spidermonkey-jsawk-resty-on-snow-leopard

jq

https://stedolan.github.io/jq/

32-bit system:

- wget http://stedolan.github.io/jq/download/linux32/jq

64-bit system:

- wget http://stedolan.github.io/jq/download/linux64/jq
- chmod +x ./jq
- sudo cp jq /usr/local/bin

Another method is to use an existing programming language typically available with your native operating system, such as Perl or Python.

```
$ which perl
/usr/bin/perl

$ which python
/usr/bin/python
```

Example: Use python to pretty format the JSON data string.

Pretty JSON using python ...

```
json='{"type":"OKResult","status":"OK","result":{"type":"SystemInfo","prod
uctType":"standard","productName":"Delphix Engine","buildTitle":"Delphix
Engine
5.1.1.0","buildTimestamp":"20160721T07:23:41.000Z","buildVersion":{"type":
"VersionInfo","major":5,"minor":1,"micro":1,"patch":0},"configured":true,"
enabedFeatures":["XPP","MSSQLHOOKS"],"apiVersion":{"type":"APIVersion","ma
jor":1,"minor":8,"micro":0},"banner":null,"locals":["enUS"],"currentLocale
":"enUS","hostname":"Delphix5110HWv8","sshPublicKey":"ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDOsrp7Aj6hFQh9yBq7273B+qtPKmCu1B18nPvr08yjt/
IZeM4qKk7caxExQS9rpfU8AWoT7e8ESV7NkBmUzOHrHnLsuJtxPqeYoqeMubVxYjJuxlH368sZ
uYsnB04KM0mi39e15lxVGvxQk9tyMpl7gs7cXRz1k6puncyiczU/axGq7ALHU2uyQoVmlPasuH
Jbq23d21VAYLuscbtgpZLAFlR8eQH5Xqaa0RT+aQJ6B1ihZ7S0ZN914M2gZHHNYcSGDWZHwUnB
Gttnxx1ofRcyN4/qwT5iHq5kjApjSaNgSAU0ExqDHiqgTq0wttf5nltCqGMTFR7XY38HiNq++a
tDroot@Delphix5110HWv8\n","memorySize":8.58107904E9,"platform":"VMware with
BIOS date
05/20/2014","uuid":"564d7e1df4cb-f91098fd348d74817683","processors":[{"typ
e":"CPUInfo","speed":2.5E9,"cores":1}],"storageUsed":2.158171648E9,"storag
eTotal":2.0673724416E10,"installationTime":"2016-07-27T13:28:46.000Z"},"jo
b":null,"action":null}'
```

Pipe the JSON data to Python programming language to pretty up the format the output for the $json string/data.

```
$ echo $json | python -mjson.tool
{
 "action": null,
 "job": null,
 "result": {
  "apiVersion": {
   "major": 1,
   "micro": 0,
   "minor": 8,
   "type": "APIVersion"
  },
  "banner": null,
  "buildTimestamp": "20160721T07:23:41.000Z",
```

```
  "buildTitle": "Delphix Engine 5.1.1.0",
  "buildVersion": {
   "major": 5,
   "micro": 1,
   "minor": 1,
   "patch": 0,
   "type": "VersionInfo"
  },
  "configured": true,
  "currentLocale": "enUS",
  "enabedFeatures": [
   "XPP",
   "MSSQLHOOKS"
 ],
 "hostname": "Delphix5110HWv8",
 "installationTime": "2016-07-27T13:28:46.000Z",
 "locals": [
  "enUS"
 ],
 "memorySize": 8581079040.0,
 "platform": "VMware with BIOS date 05/20/2014",
 "processors": [
  {
   "cores": 1,
   "speed": 2500000000.0,
   "type": "CPUInfo"
  }
 ],
 "productName": "Delphix Engine",
 "productType": "standard",
 "sshPublicKey": "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDOsrp7Aj6hFQh9yBq7273B+qtPKmCu1B18nPvr08yjt/
IZeM4qKk7caxExQS9rpfU8AWoT7e8ESV7NkBmUzOHrHnLsuJtxPqeYoqeMubVxYjJuxlH368sZ
uYsnB04KM0mi39e15lxVGvxQk9tyMpl7gs7cXRz1k6puncyiczU/axGq7ALHU2uyQoVmlPasuH
Jbq23d21VAYLuscbtgpZLAFlR8eQH5Xqaa0RT+aQJ6B1ihZ7S0ZN914M2gZHHNYcSGDWZHwUnB
Gttnxx1ofRcyN4/qwT5iHq5kjApjSaNgSAU0ExqDHiqgTq0wttf5nltCqGMTFR7XY38HiNq++a
tDroot@Delphix5110HWv8\n",
 "storageTotal": 20673724416.0,
 "storageUsed": 2158171648.0,
 "type": "SystemInfo",
 "uuid": "564d7e1df4cb-f91098fd348d74817683"
 },
 "status": "OK",
```

```
 "type": "OKResult"
}
```

jq parser

The jq command line parser is available on Unix, Linux, Mac, and Windows platforms. Typically, for Windows, the built-in ConvertFrom/To-Json object parser will be used. "jq" is being included in most native Linux distributions and is easy to install on the Mac OS.

References:

http://www.compciv.org/recipes/cli/jq-for-parsing-json/

https://github.com/stedolan/jq/wiki/FAQ#installation

Mac Installation: http://macappstore.org/jq/

Example:

```
json='{"type":"ListResult","status":"OK","result":[{"type":"OracleLinkedSo
urce","reference":"ORACLE_LINKED_SOURCE-52","namespace":null,"name":"DPXDE
V01","description":null,"virtual":false,"restoration":false,"staging":fals
e,"container":"ORACLE_DB_CONTAINER-120","config":"ORACLE_SINGLE_CONFIG-40"
,"status":"DEFAULT","runtime":{"type":"OracleSourceRuntime","status":"RUNN
ING","accessible":true,"databaseSize":2.409529344E9,"notAccessibleReason":
null,"databaseMode":"READ_WRITE","lastNonLoggedLocation":"0","activeInstan
ces":[{"type":"OracleActiveInstance","instanceNumber":1,"instanceName":"DP
XDEV01","hostName":"linuxtarget.delphix.local"}],"databaseStats":null,"bct
Enabled":true,"racEnabled":null,"dnfsEnabled":false,"archivelogEnabled":nu
ll},"backupLevelEnabled":false,"rmanChannels":2,"filesPerSet":5,"checkLogi
cal":false,"externalFilePath":null,"encryptedLinkingEnabled":false,"compre
ssedLinkingEnabled":true,"bandwidthLimit":0,"numberOfConnections":1,"enabl
ed":true,"preScript":"","postScript":"","role":"PRIMARY"},{"type":"OracleV
irtualSource","reference":"ORACLE_VIRTUAL_SOURCE-25","namespace":null,"nam
e":"VBITT","description":null,"virtual":true,"restoration":false,"staging"
:false,"container":"ORACLE_DB_CONTAINER-121","config":"ORACLE_SINGLE_CONFI
G-47","status":"DEFAULT","runtime":{"type":"OracleSourceRuntime","status":
"RUNNING","accessible":true,"databaseSize":2.410053632E9,"notAccessibleRea
son":null,"databaseMode":"READ_WRITE","lastNonLoggedLocation":"0","activeI
nstances":[{"type":"OracleActiveInstance","instanceNumber":1,"instanceName
":"VBITT","hostName":"linuxtarget.delphix.local"}],"databaseStats":[{"type
":"OracleDatabaseStatsSection","sectionName":"Open
Transactions","columnHeaders":["Transaction
Count"],"rowValues":[{"type":"OracleDatabaseStatistic","statisticValues":[
"0"]}]},{"type":"OracleDatabaseStatsSection","sectionName":"Session
Statistics","columnHeaders":["Current Session","Total Session","High
Watermark"],"rowValues":[{"type":"OracleDatabaseStatistic","statisticValue
s":["2","46","5"]}]},{"type":"OracleDatabaseStatsSection","sectionName":"T
op Wait Events","columnHeaders":["Event","Wait Count","Total Wait Time
(s)"],"rowValues":[{"type":"OracleDatabaseStatistic","statisticValues":["D
isk file operations
I/O","13","13"]},{"type":"OracleDatabaseStatistic","statisticValues":["log
file sequential
read","11","12"]},{"type":"OracleDatabaseStatistic","statisticValues":["co
```

```
ntrol file parallel
write","8","8"]},{"type":"OracleDatabaseStatistic","statisticValues":["con
trol file sequential
read","6","3"]},{"type":"OracleDatabaseStatistic","statisticValues":["ARCH
wait for process start
3","2","2"]},{"type":"OracleDatabaseStatistic","statisticValues":["db file
sequential
read","9","1"]},{"type":"OracleDatabaseStatistic","statisticValues":["rdbms
ipc
reply","1","1"]},{"type":"OracleDatabaseStatistic","statisticValues":["JS
coord start
wait","1","1"]},{"type":"OracleDatabaseStatistic","statisticValues":["os
thread
startup","2","0"]},{"type":"OracleDatabaseStatistic","statisticValues":["P
arameter File
I/O","1","0"]}]},{"type":"OracleDatabaseStatsSection","sectionName":"Top
SQL by CPU","columnHeaders":["Percentage of Load","SQL
Statement"],"rowValues":[]}],"bctEnabled":false,"racEnabled":null,"dnfsEna
bled":false,"archivelogEnabled":null},"operations":{"type":"VirtualSourceO
perations","configureClone":[],"preRefresh":[],"postRefresh":[]},"mountBas
e":"/mnt/provision","fileMappingRules":null,"manualProvisioning":null,"con
figParams":{"memory_target":"1191182336","processes":"150","log_archive_de
st_1":"location=/mnt/provision/VBITT/archive/
MANDATORY","_omf":"ENABLED","filesystemio_options":"setall","compatible":"
11.2.0.4.0","audit_trail":"NONE","remote_login_passwordfile":"EXCLUSIVE","
```

```
open_cursors":"300","audit_sys_operations":"FALSE"},"configTemplate":null,
"nodeListenerList":[],"enabled":true,"role":"PRIMARY"}],"job":null,"action
":null,"total":2,"overflow":false}'
```

We have a very big JSON string above. Let's perform some basic jq parsing.

1. Pipe JSON string into jq command line parser.

   ```
   echo $json | jq '.'
   ```

   The output is pretty human-readable JSON formatted string.
2. Get the first-level status value (. . . ,"status":"OK", . . .)

   ```
   echo $json | jq '.status'
   "OK"
   ```

3. Get raw values (not quoted).

   ```
   echo $json | jq --raw-output '.status'
   OK
   ```

4. Get number of rows returned for the type equal to "ListResult" API returned request.

   ```
   echo $json | jq --raw-output '.total'
   2
   ```

5. Get first result set.

```
echo $json | jq '.result[0] '
{
 "type": "OracleLinkedSource",
 "reference": "ORACLE_LINKED_SOURCE-52",
 "namespace": null,
 "name": "DPXDEV01",
 "description": null,
 "virtual": false,
 "restoration": false,
 "staging": false,
 "container": "ORACLE_DB_CONTAINER-120",
 "config": "ORACLE_SINGLE_CONFIG-40",
 "status": "DEFAULT",
 "runtime": {
  "type": "OracleSourceRuntime",
  "status": "RUNNING",
  "accessible": true,
  "databaseSize": 2409529344,
  "notAccessibleReason": null,
  "databaseMode": "READ_WRITE",
  "lastNonLoggedLocation": "0",
  "activeInstances": [
   {
    "type": "OracleActiveInstance",
    "instanceNumber": 1,
    "instanceName": "DPXDEV01",
    "hostName": "linuxtarget.delphix.local"
   }
  ],
  "databaseStats": null,
  "bctEnabled": true,
  "racEnabled": null,
  "dnfsEnabled": false,
  "archivelogEnabled": null
 },
 "backupLevelEnabled": false,
 "rmanChannels": 2,
 "filesPerSet": 5,
 "checkLogical": false,
 "externalFilePath": null,
 "encryptedLinkingEnabled": false,
 "compressedLinkingEnabled": true,
 "bandwidthLimit": 0,
 "numberOfConnections": 1,
 "enabled": true,
 "preScript": "",
 "postScript": "",
 "role": "PRIMARY"
}
```

6. Get first result set name value.

```
echo $json | jq --raw-output '.result[0].name'
DPXDEV01
```

7. Get first result set reference value.

```
echo $json | jq --raw-output '.result[0].reference'
```

8. Get first result set name=value pairs.

```
echo $json | jq '.result[0]' | jq -r
"to_entries|map(\"(.key)=(.value|tostring)\")|.[]" | grep container


container=ORACLE_DB_CONTAINER-120
```

9. Get ALL result sets name values.

```
echo $json | jq '.result[].name'
"DPXDEV01"
"VBITT"
```

10. Get ALL result sets "reference" and "container" values.

```
echo $json | jq '.result[].reference,.result[].container'
"ORACLE_LINKED_SOURCE-52"
"ORACLE_VIRTUAL_SOURCE-25"
"ORACLE_DB_CONTAINER-120"
"ORACLE_DB_CONTAINER-121"
```

11. Now, let's scan ALL result sets for a conditional match and return a related value.

```
echo $json | jq --raw-output '.result[] | select(.name=="VBITT") |
.container'
ORACLE_DB_CONTAINER-121

echo $json | jq --raw-output '.result[] | select(.name=="VBITT") |
.reference'
ORACLE_VIRTUAL_SOURCE-25

echo $json | jq --raw-output '.result[] | select(.name=="VBITT") |
.container, .reference'
ORACLE_DB_CONTAINER-121
ORACLE_VIRTUAL_SOURCE-25
```

This is a typical usage for Delphix, where the human readable name is provided and we need to look up the object reference, container, status, etc. for the respective name. Some object references are based on an expressions such as "and" or "or" conditions.

```
echo $json | jq --raw-output '.result[] |
select(.environment=="UNIX_HOST_ENVIRONMENT-9" and
.name=="/u02/ora/app/product/11.2.0/dbhome_1" ) | .reference '
```

In this case, the jq select command has an "and" condition in order to correctly identify the target result object index. This is important for getting the correct and single return value for `| .reference`, since there might be more than one instance within the environment.

For a working example of using the jq JSON parser, see the VDB Init using jq Command Line JSON Parser use case, **Filename: vdb_init.sh**. A version of all the Unix/Linux/Mac shell scripts exists within the code provided. It contains the ***_jq.sh** within the filename.

PowerShell

Starting with Powershell 3.0, there are ConvertFrom-Json and ConvertTo-Json modules/commands to parse the JSON string data to/from objects. If you are stuck with Powershell 2.x., the next section provides similar functions as a method of working with JSON strings.

> These 2.x functions are not 100% the same as the Powershell 3.0 ConvertFrom-Json/ConvertTo-Json modules.

PowerShell 2 Example

Filename: parse_2.0_ps1.txt (rename to parse_2.0.ps1)

For Powershell 2.0, there are no JSON-provided functions or commands, so the following will serialize the JSON data to a serialized array.

```
function ConvertTo-Json20([object] $item){
 add-type -assembly system.web.extensions
 $ps_js=new-object system.web.script.serialization.javascriptSerializer
 return $ps_js.Serialize($item)
}


function ConvertFrom-Json20([object] $item){
 add-type -assembly system.web.extensions
 $ps_js=new-object system.web.script.serialization.javascriptSerializer
 #The comma operator is the array construction operator in PowerShell
return ,$ps_js.DeserializeObject($item)
}
```

Use the JSON from the system API Call.

```
$json='{"type":"OKResult","status":"OK","result":{"type":"SystemInfo","pro
ductType":"standard","productName":"Delphix Engine","buildTitle":"Delphix
Engine
5.1.1.0","buildTimestamp":"20160721T07:23:41.000Z","buildVersion":{"type":
"VersionInfo","major":5,"minor":1,"micro":1,"patch":0},"configured":true,"
enabedFeatures":["XPP","MSSQLHOOKS"],"apiVersion":{"type":"APIVersion","ma
jor":1,"minor":8,"micro":0},"banner":null,"locals":["enUS"],"currentLocale
":"enUS","hostname":"Delphix5110HWv8","sshPublicKey":"ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDOsrp7Aj6hFQh9yBq7273B+qtPKmCu1B18nPvr08yjt/
IZeM4qKk7caxExQS9rpfU8AWoT7e8ESV7NkBmUzOHrHnLsuJtxPqeYoqeMubVxYjJuxlH368sZ
uYsnB04KM0mi39e15lxVGvxQk9tyMpl7gs7cXRz1k6puncyiczU/axGq7ALHU2uyQoVmlPasuH
Jbq23d21VAYLuscbtgpZLAFlR8eQH5Xqaa0RT+aQJ6B1ihZ7S0ZN914M2gZHHNYcSGDWZHwUnB
Gttnxx1ofRcyN4/qwT5iHq5kjApjSaNgSAU0ExqDHiqgTq0wttf5nltCqGMTFR7XY38HiNq++a
tDroot@Delphix5110HWv8\n","memorySize":8.58107904E9,"platform":"VMware with
BIOS date
05/20/2014","uuid":"564d7e1df4cb-f91098fd348d74817683","processors":[{"typ
e":"CPUInfo","speed":2.5E9,"cores":1}],"storageUsed":2.158171648E9,"storag
eTotal":2.0673724416E10,"installationTime":"2016-07-27T13:28:46.000Z"},"jo
b":null,"action":null}'
```

Convert the JSON string.

> The job and action are null values.

```
PS> $o = ConvertFrom-Json20 $json
PS> $o

Key    Value
---    -----
type   OKResult
status   OK
result   {[type, SystemInfo], [productType, standard], [productNa...
job
action
```

Extract the result JSON string array.

```
PS> $a = $o.result
PS> $a

Key     Value
---     -----
type    SystemInfo
productType   standard
productName   Delphix Engine
buildTitle   Delphix Engine 5.1.1.0
buildTimestamp  20160721T07:23:41.000Z
buildVersion  {[type, VersionInfo], [major, 5], [minor, 1], [micro, 1]...
configured   True
enabedFeatures  {XPP, MSSQLHOOKS}
apiVersion   {[type, APIVersion], [major, 1], [minor, 8], [micro, 0]}
banner
locals    {enUS}
currentLocale  enUS
hostname   Delphix5110HWv8
sshPublicKey  ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDOsrp7Aj6hFQh9yBq7...
memorySize   8581079040
platform   VMware with BIOS date 05/20/2014
uuid    564d7e1df4cb-f91098fd348d74817683
processors   {System.Collections.Generic.Dictionary`2[System.String,S...
storageUsed   2158171648
storageTotal  20673724416
installationTime 2016-07-27T13:28:46.000Z


foreach ($element in $a) {$element}
```

same output as above

```
PS> $a.type
SystemInfo
PS> $a.buildTitle
Delphix Engine 5.1.1.0
PS> $a.hostname
Delphix5110HWv8
```

Extract the result.buildVersion object.

```
PS> $a1 = $o.result.buildVersion
PS> $a1

Key     Value
---     -----
type    VersionInfo
major    5
minor    1
micro    1
patch    0

PS> $a1.major
5
```

Extract the result.processors array collection.

```
PS> $b = $o.result.processors
PS> $b

Key      Value
---      -----
type     CPUInfo
speed     2500000000
cores     1


PS> $a -is [Array]
False
PS> $a -is [Object]
True
PS> $b -is [Array]
True
```

Convert Array Collection to Object.

```
PS> $b1 = $b | Select-Object
PS> $b1

Key     Value
---     ------
type    CPUInfo
speed    2500000000
cores    1


PS> $b1.type
CPUInfo
PS> $b1.speed
2500000000
```

PowerShell 3 or greater Example

Starting with Powershell 3.0, there is are ConvertFrom-Json and ConvertTo-Json commands to parse the JSON data to/from objects.

Reference: https://technet.microsoft.com/en-us/library/hh849898(v=wps.620).aspx

```
$o = $json | ConvertFrom-Json
```

There are a number of tutorials and functional examples on the web. Below is an excerpt from the Powershell introduction video for Linux / Mac Open Source announcement.

https://youtu.be/2WZwv7TxqZ0

Powershell JSON ConvertTo-Json and Python Example 15:55 through 21:16

The concept is straightforward:

- The `ConvertFrom-Json` JSON string is converted into a Powershell object that you can reference directly.
- The `ConvertTo-Json` takes the JSON object and converts it to a string.

JSON Parsing from within Programming Languages

Most programming languages provide their own libraries, functions, and methods for parsing JSON data strings into objects/hashes/arrays/xml that the native programming language can easily process.

API Use Case Commands and Scripts

Sample Script Parsers

The Delphix Use Cases scripts provided use the native operating system "curl" command and then a JSON parser, depending on the engine on which you are runnning the scripts.

For Unix/Linux/Mac, the scripts provided use both native shell commands and/or the jq parser program commands. Subroutines have been provided for both methods:

```
Native Shell commands:  parseJSON_subroutines.sh
jq Parser commands:   jqJSON_subroutines.sh
```

For Windows, the scripts are for Powershell 2.0 and utilitze the custom `ConvertFrom-Json20` and `ConvertTo-Json20` functions provided. As noted, with Powershell 3.0, there are `ConvertFrom-Json` and `ConvertTo-Json` command-lets provided by Powershell.

Using the jq Parser

These are some of the jq commands used within the scripts. The first is a shell script subroutine which is used for finding and returning a single item value.

Filename: jqJSON_subroutines.sh

Subroutines

This code requires the jq Linux/Mac JSON parser program

```
jqParse() {
 STR=$1     # json string
 FND=$2     # name to find
 RESULTS=""    # returned name value
 RESULTS=`echo $STR | jq --raw-output '.'"$FND"''`
 #echo "Results: ${RESULTS}"
 if [ "${FND}" == "status" ] && [ "${RESULTS}" != "OK" ]
 then
  echo "Error: Invalid Satus, please check code ... ${STR}"
  exit 1;
 elif [ "${RESULTS}" == "" ]
 then
  echo "Error: No Results ${FND}, please check code ... ${STR}"
  exit 1;
 fi
 echo "${RESULTS}"
}
```

The subroutine is called from the shell script to return values based on the key (or name) value provided.

Usage – After every curl command, check that the returned status value is "OK".

```
RESULTS=$( jqParse "${STATUS}" "status" )
```

Call the jqParse subroutine, where

- ${STATUS} is the returned JSON string from the curl command, and
- the value we want returned is where the name/key is equal to "status"

Usage – Get a single value within the returned nested result object:

```
JOBSTATE=$( jqParse "${JOB_STATUS}" "result.jobState" )
```

Call the jqParse subroutine, where

- ${JOB_STATUS} is the returned JSON string from the cURL command, and
- the value we want returned is where the name/key is equal to "jobState" with the nested ".result" object.

Usage – Find name/value result object and return another value within the select result object:

Use jq to parse out container reference for name of $SOURCE_SID ...

```
CONTAINER_REFERENCE=`echo ${STATUS} | jq --raw-output '.result[] |
select(.name=="'"${SOURCE_SID}"'") | .reference '`
```

where

- ${STATUS} is the returned JSON string from the cURL command, and
- the value we want returned is based on the selected nested result object where the .result[].name is equal to "${SOURCE_SID}" and return the .reference value for the selected result object.

Delphix Engine Use Cases

Delphix User Session Timeout

Some activities can take longer than the default 30 minute session timeout value. Therefore, the following script allows you to change the timeout value using the RESTFul API. As always, you can change it easily through the CLI.

> This code is the first example showing how object references are used for input (either JSON or URL) into API calls. The name will be the **DE_USER** variable value **delphix_admin**. The object reference that the code identifies is **USER-2**, which in this case is passed into the API URL to update the user parameters passed via the JSON string.

Filename: user_timeout.sh

Edit the file to update the parameters as required for your environment.

```
####################
# DELPHIX CORP #
####################
#Parameter Initialization
DMIP=172.16.160.195
#DMPORT=8282
DMUSER=delphix_admin
DMPASS=delphix
COOKIE="~/cookies.txt"
COOKIE=`eval echo $COOKIE`
CONTENT_TYPE="Content-Type: application/json"
BaseURL="http://${DMIP}/resources/json/delphix"
#
Required for user timeout ...
#
DE_USER="delphix_admin" # Delphix Engine User
DE_TIMEOUT=120 # Timeout integer in minutes
##############################################
# NO CHANGES REQUIRED BELOW THIS POINT #
##############################################
```

Sample Output

```
$ ./user_timeout.sh # or ./user_timeout_jq.sh
Authenticating on http://172.16.160.195/resources/json/delphix
Session and Login Successful ...
user reference: USER-2
Update delphix_admin session timeout value to 120 minutes ...
Returned JSON:
{"type":"OKResult","status":"OK","result":"","job":null,"action":"ACTION-4
23"}
Results: OK
Done ...
$
```

VDB Init (start | stop | enable | disable | status | delete)

This script is used to start, stop, enable, disable, and delete a Delphix platform source object. Typically, this is done on a virtual databases (VDBs), but you can use it for dSources as well.

> The `vdb_init.sh` and `vdb_operations.sh` require the "jq" command line json parser.

Filename: vdb_init.sh

The `vdb_init.sh` supports the start, stop, enable, disable, status, and delete command line options.

```
$ ./vdb_init.sh something VBITT
. . .
Unknown option (start | stop | enable | disable | status | delete):
something
Exiting ...
$ ./vdb_init.sh status VBITT
database container reference: ORACLE_DB_CONTAINER-121
source reference: ORACLE_VIRTUAL_SOURCE-25
Runtime Status: "INACTIVE"
Enabled: true
Done ...
```

```
$ ./vdb_init.sh start VBITT
database container reference: ORACLE_DB_CONTAINER-121
source reference: ORACLE_VIRTUAL_SOURCE-25
Job: JOB-894
Current status as of Wed Sep 7 16:04:04 EDT 2016 : RUNNING 0% Completed
Current status as of Wed Sep 7 16:04:14 EDT 2016 : RUNNING 25% Completed
Current status as of Wed Sep 7 16:04:24 EDT 2016 : RUNNING 45% Completed
Job: JOB-894 COMPLETED 100% Completed ...
Done ...
```

```
$ ./vdb_init.sh delete VBITT
database container reference: ORACLE_DB_CONTAINER-123
source reference: ORACLE_VIRTUAL_SOURCE-27
vendor source: OracleVirtualSource
delete parameters type: OracleDeleteParameters
Job: JOB-927
Current status as of Sat Sep 10 12:55:32 EDT 2016 : RUNNING 0% Completed
Current status as of Sat Sep 10 12:55:32 EDT 2016 : RUNNING 0% Completed
Job: JOB-927 COMPLETED 100% Completed ...
Done ...
```

VDB Operations (sync, refresh, rollback)

This script is used to perform a sync (snapshot), refresh, or rollback (reset) on the Delphix Engine source object. All these work on a virtual databases (VDBs), but only a sync operation can be used on dSources.

> The `vdb_init.sh` and `vdb_operations.sh` require the "jq" command line json parser.

Filename: vdb_operations.sh

```
$ ./vdb_operations.sh sync VBITT
Session and Login Successful ...
database container reference: ORACLE_DB_CONTAINER-131
provision source container: ORACLE_DB_CONTAINER-129
json> {
 "type": "OracleSyncParameters"
}
Job: JOB-998
Current status as of Wed Sep 14 17:05:00 EDT 2016 : RUNNING 0% Completed
```

```
Current status as of Wed Sep 14 17:05:24 EDT 2016 : RUNNING 97% Completed
Job: JOB-998 COMPLETED 100% Completed ...
Done ...
```

Rollback rewinds the virtual database back to the last point in time within the source TimeFlow.

```
$ ./vdb_operations.sh rollback VBITT
Session and Login Successful ...
database container reference: ORACLE_DB_CONTAINER-131
provision source container: ORACLE_DB_CONTAINER-129
json> {
 "type": "OracleRollbackParameters",
 "timeflowPointParameters": {
  "type": "TimeflowPointSemantic",
  "container": "ORACLE_DB_CONTAINER-131"
 }
}
Job: JOB-1000
Current status as of Wed Sep 14 17:06:33 EDT 2016 : RUNNING 0% Completed
Current status as of Wed Sep 14 17:06:43 EDT 2016 : RUNNING 0% Completed
Current status as of Wed Sep 14 17:06:53 EDT 2016 : RUNNING 34% Completed
Current status as of Wed Sep 14 17:07:03 EDT 2016 : RUNNING 34% Completed
Current status as of Wed Sep 14 17:07:13 EDT 2016 : RUNNING 58% Completed
Current status as of Wed Sep 14 17:07:23 EDT 2016 : RUNNING 67% Completed
Current status as of Wed Sep 14 17:07:33 EDT 2016 : RUNNING 70% Completed
Current status as of Wed Sep 14 17:07:43 EDT 2016 : RUNNING 70% Completed
Current status as of Wed Sep 14 17:07:53 EDT 2016 : RUNNING 71% Completed
Current status as of Wed Sep 14 17:08:03 EDT 2016 : RUNNING 72% Completed
Current status as of Wed Sep 14 17:08:13 EDT 2016 : RUNNING 73% Completed
Current status as of Wed Sep 14 17:08:23 EDT 2016 : RUNNING 94% Completed
Current status as of Wed Sep 14 17:08:33 EDT 2016 : RUNNING 96% Completed
Current status as of Wed Sep 14 17:08:43 EDT 2016 : RUNNING 96% Completed
Current status as of Wed Sep 14 17:08:53 EDT 2016 : RUNNING 96% Completed
Job: JOB-1000 COMPLETED 100% Completed ...
Done ...
```

Refresh recreates the virtual database to the last point in time in the respective parent, provision source TimeFlow.

```
$ ./vdb_operations.sh refresh VBITT
Session and Login Successful ...
database container reference: ORACLE_DB_CONTAINER-131
provision source container: ORACLE_DB_CONTAINER-129
json> {
 "type": "OracleRefreshParameters",
 "timeflowPointParameters": {
  "type": "TimeflowPointSemantic",
  "container": "ORACLE_DB_CONTAINER-129"
 }
}
Job: JOB-1005
Current status as of Wed Sep 14 17:10:11 EDT 2016 : RUNNING 0% Completed
Current status as of Wed Sep 14 17:10:21 EDT 2016 : RUNNING 0% Completed
Current status as of Wed Sep 14 17:10:31 EDT 2016 : RUNNING 34% Completed
Current status as of Wed Sep 14 17:10:41 EDT 2016 : RUNNING 35% Completed
Current status as of Wed Sep 14 17:10:51 EDT 2016 : RUNNING 70% Completed
Current status as of Wed Sep 14 17:11:01 EDT 2016 : RUNNING 70% Completed
Current status as of Wed Sep 14 17:11:20 EDT 2016 : RUNNING 72% Completed
Current status as of Wed Sep 14 17:11:31 EDT 2016 : RUNNING 73% Completed
Current status as of Wed Sep 14 17:11:41 EDT 2016 : RUNNING 73% Completed
Job: JOB-1005 COMPLETED 100% Completed ...
Done ...
```

API Analytics Use Cases

As stated earlier, Delphix provides a toolkit, dxtoolkit2, that already performs a number of typical functionalities. The utility ***dx_get_analytics*** is absolutely great for dumping analytic data into a .csv (comma separated value) format from the Delphix Engine, which you can then easily integrate into your enterprise monitoring tools. For the dxtoolkit2 README document table of contents, see the Appendix.

dxtoolkit2 "db_get_analytics" Example

First, get the latest dxtoolkit2 from your Delphix Pre-Sales personnel for your respective platform. The tool is easily configured through the **dxtools.conf** json formatted file.

> You can encrypt the password. See respective documentation for instructions.

```
[dxtoolkit2]$ more dxtools.conf
{
 "data":[
 {
  "hostname" : "linuxtarget",
  "ip_address" : "172.16.160.164",
  "username" : "delphix_admin",
  "password" : "delphix",
  "port" : "80",
  "default" : "true"
 },
 {
  "hostname" : "DelphixEngine",
  "ip_address" : "172.16.160.195",
  "username" : "delphix_admin",
  "password" : "delphix",
  "port" : "80",
  "default" : "true",
  "encrypted" : "false"
 }
 ]
}
```

Verify/set the file permissions.

Run the **dx_get_analytics** command.

```
[dxtoolkit2]$ chmod +x dx*
[dxtoolkit2]$ ./dx_get_analytics -d DelphixEngine -i 3600 -t all -outdir
/tmp
Connected to Delphix Engine DelphixEngine (IP 172.16.160.195)

Generating cpu raw report file /tmp/DelphixEngine-analytics-cpu-raw.csv
Generating cpu aggregated report file
/tmp/DelphixEngine-analytics-cpu-aggregated.csv
Generating disk raw report file /tmp/DelphixEngine-analytics-disk-raw.csv
Generating disk aggregated report file
/tmp/DelphixEngine-analytics-disk-aggregated.csv
Generating iscsi raw report file /tmp/DelphixEngine-analytics-iscsi-raw.csv
Generating iscsi aggregated report file
/tmp/DelphixEngine-analytics-iscsi-aggregated.csv
Generating network raw report file
/tmp/DelphixEngine-analytics-network-raw.csv
Generating network aggregated report file
/tmp/DelphixEngine-analytics-network-aggregated.csv
Generating nfs raw report file /tmp/DelphixEngine-analytics-nfs-raw.csv
Generating nfs aggregated report file
/tmp/DelphixEngine-analytics-nfs-aggregated.csv
Generating tcp raw report file /tmp/DelphixEngine-analytics-tcp-raw.csv
Generating tcp aggregated report file
/tmp/DelphixEngine-analytics-tcp-aggregated.csv
[dxtoolkit2]$
```

dx_get_analytics Generated (.csv) Files

```
[dxtoolkit2]$ cd /tmp
[dxtoolkit2]$ ls –ltr *.csv

rw-r r- 1 delphix oinstall 1141 Aug 11 11:25
DelphixEngine-analytics-cpu-raw.csv
rw-r r- 1 delphix oinstall 160  Aug 11 11:25
DelphixEngine-analytics-cpu-aggregated.csv
rw-r r- 1 delphix oinstall 2127 Aug 11 11:25
DelphixEngine-analytics-disk-raw.csv
rw-r r- 1 delphix oinstall 511  Aug 11 11:25
DelphixEngine-analytics-disk-aggregated.csv
rw-r r- 1 delphix oinstall 124  Aug 11 11:25
DelphixEngine-analytics-iscsi-raw.csv
rw-r r- 1 delphix oinstall 260  Aug 11 11:25
DelphixEngine-analytics-iscsi-aggregated.csv
rw-r r- 1 delphix oinstall 1206  Aug 11 11:25
DelphixEngine-analytics-network-raw.csv
rw-r r- 1 delphix oinstall 275  Aug 11 11:25
DelphixEngine-analytics-network-aggregated.csv
rw-r r- 1 delphix oinstall 88   Aug 11 11:25
DelphixEngine-analytics-nfs-raw.csv
rw-r r- 1 delphix oinstall 260  Aug 11 11:25
DelphixEngine-analytics-nfs-aggregated.csv
rw-r r- 1 delphix oinstall 3258  Aug 11 11:25
DelphixEngine-analytics-tcp-raw.csv
rw-r r- 1 delphix oinstall 494  Aug 11 11:25
DelphixEngine-analytics-tcp-aggregated.csv
```

Sample .csv File Content Generated

```
[dxtoolkit2]$ more DelphixEngine-analytics-tcp-aggregated.csv
#time,client,inBytes_min,inBytes_max,inBytes_85pct,outBytes_min,outBytes_m
ax,outBytes_85pct
2016-08-05,172.16.160.183-3260-49228,0.00,2.00,2.00,0.00,2.00,2.00
2016-08-06,172.16.160.183-3260-49228,0.00,2.00,2.00,0.00,2.00,2.00
2016-08-07,172.16.160.183-3260-49228,2.00,2.00,2.00,2.00,2.00,2.00
2016-08-08,172.16.160.183-3260-49228,2.00,2.00,2.00,2.00,2.00,2.00
2016-08-08,172.16.160.183-3260-57658,1.00,1.00,1.00,1.00,1.00,1.00
2016-08-08,172.16.160.183-3260-57745,2.00,2.00,2.00,2.00,2.00,2.00
```

Delphix Self-Service Use Cases for APIs

- Create Delphix Self-Service Template
- Create Delphix Self-Service Data Container
- Create Delphix Self-Service Bookmark
- Delphix Self-Service Refresh

Create Delphix Self-Service Template

> Jet Stream is now known as Delphix Self-Service.

Filename: jetstream_template.sh or *jetstream_template_jq.sh*

Edit the file to update the parameters as required for your environment.

Required for Delphix Self-Service Template ...

```
TPL_NAME="jstpl"   # JetStream Template Name
DATASOURCE_NAME="jsds"  # JetStream Data Source Name
DATASOURCE_VDB="VBITT"  # JetStream Data Source VDB or dSource
```

Sample Output

```
$ ./jetstream_template.sh# or ./Jetstream_template_jq.sh
Authenticating on http://172.16.160.195/resources/json/delphix
Session and Login Successful ...
Getting Database Container Reference Value ...
container reference: ORACLE_DB_CONTAINER-45
Create JetStream Template jstpl with Data Source DB VBITT ...
Database:
{"type":"OKResult","status":"OK","result":"JS_DATA_TEMPLATE-3","job":null,
"action":"ACTION-547"}


Done ... (no job required for this action)
```

Create Delphix Self-Service Data Container

Filename: jetstream_container.sh# or *jetstream_container_jq.sh*

Edit the file to update the parameters as required for your environment.

Required for Delphix Self-Service Container ...

```
TPL_NAME="jstpl"  # JetStream Template Name
DS_NAME="jsds"   # JetStream Data Source Name


DC_NAME="jsdc"   # JetStream Data Container Name
DC_VDB="VBITT2"   # JetStream Data Container VDB
```

Sample Output

```
$ ./jetstream_container.sh# or ./jetstream_container_jq.sh
Authenticating on http://172.16.160.195/resources/json/delphix
Session and Login Successful ...
Getting Database Container Reference Value ...
container reference: ORACLE_DB_CONTAINER-46
JetStream Data Template: JS_DATA_TEMPLATE-4
JetStream sourceDataLayout: JS_DATA_TEMPLATE-4
Create JetStream Container jsdc with Data Source DB VBITT2 ...
JetStream Data Container Creation Results:
{"type":"OKResult","status":"OK","result":"JS_DATA_CONTAINER-4","job":"JOB
-240","action":"ACTION-569"}
Job: JOB-240
Current status as of Wed Aug 17 04:11:54 EDT 2016 : RUNNING 0.0% Completed
Current status as of Wed Aug 17 04:11:54 EDT 2016 : RUNNING 0.0% Completed
Current status as of Wed Aug 17 04:12:04 EDT 2016 : RUNNING 5.0% Completed
Current status as of Wed Aug 17 04:12:14 EDT 2016 : RUNNING 5.0% Completed
Current status as of Wed Aug 17 04:12:24 EDT 2016 : RUNNING 30.0% Completed
Current status as of Wed Aug 17 04:12:34 EDT 2016 : RUNNING 31.0% Completed
Current status as of Wed Aug 17 04:12:44 EDT 2016 : RUNNING 53.0% Completed
Current status as of Wed Aug 17 04:12:54 EDT 2016 : RUNNING 57.0% Completed
Current status as of Wed Aug 17 04:13:04 EDT 2016 : RUNNING 57.0% Completed
Current status as of Wed Aug 17 04:13:14 EDT 2016 : RUNNING 57.0% Completed
Current status as of Wed Aug 17 04:13:24 EDT 2016 : RUNNING 59.0% Completed
Current status as of Wed Aug 17 04:13:34 EDT 2016 : RUNNING 60.0% Completed
Current status as of Wed Aug 17 04:13:44 EDT 2016 : RUNNING 77.0% Completed
Current status as of Wed Aug 17 04:13:54 EDT 2016 : RUNNING 77.0% Completed
Current status as of Wed Aug 17 04:14:04 EDT 2016 : RUNNING 77.0% Completed
Current status as of Wed Aug 17 04:14:14 EDT 2016 : RUNNING 77.0% Completed
Job: JOB-240 COMPLETED 100.0% Completed ...


Done ...
```

Create Delphix Self-Service Bookmark

Filename: jetstream_api_examples.txt (part 1)

Create Bookmark ...

Change parameters as required and desired.

```
curl -X POST -k --data @-
http://172.16.160.177/resources/json/delphix/jetstream/bookmark \
 -b cookies.txt -H "Content-Type: application/json" <<EOF
{
 "type": "JSBookmarkCreateParameters",
 "bookmark": {
  "type": "JSBookmark",
  "name": "aalen",
  "branch": "JS_BRANCH-5",
  "shared": false,
  "tags": [
   "A",
   "B",
   "C"
  ]
 },
 "timelinePointParameters": {
  "type": "JSTimelinePointLatestTimeInput",
  "sourceDataLayout": "JS_DATA_CONTAINER-2"
 }
}
EOF
{"type":"OKResult","status":"OK","result":"JS_BOOKMARK-5","job":"JOB-512",
"action":"ACTION-921"}
```

The timelinePointParameters type "JSTimelinePointLatestTimeInput" is the last point / latest time in the branch!

Filename: jetstream_bookmark.sh or *jetstream_bookmark_jq.sh*

Edit the file to update the parameters as required for your environment.

```
DT=`date '+%Y%m%d%H%M%S'`
```

Required for Delphix Self-Service Bookmark ...

```
JS_BRANCH="default"   # JetStream Branch
BM_NAME="aalen_${DT}"  # JetStream Bookmark Name appended timestamp
SHARED="false"    # Share Bookmark true/false
TAGS='"API","Created"'  # Tags Array Values
```

Sample Output

```
$ ./jetstream_bookmark.sh # or ./jetstream_bookmark_jq.sh
Authenticating on http://172.16.160.195/resources/json/delphix
Session and Login Successful ...
Getting Jetstream Branch Reference Value ...
branch reference: JS_BRANCH-7
dataLayout container reference: JS_DATA_CONTAINER-4
JetStream Bookmark Creation Results:
{"type":"OKResult","status":"OK","result":"JS_BOOKMARK-4","job":"JOB-251",
"action":"ACTION-591"}
Job: JOB-251
Current status as of Wed Aug 17 04:59:53 EDT 2016 : COMPLETED 100.0%
Completed
Job: JOB-251 COMPLETED 100.0% Completed ...


Done ...
```

Delphix Self-Service Refresh

Filename: jetstream_api_examples.txt (part 2)

Use CLI command to get Delphix Self-Service Container Reference

```
/jetstream/container/list

   ...
   "reference": " JS_DATA_CONTAINER-4 ",
   "namespace": null,
   "name": " jsdc ",
   ...
```

Refresh Container Information ...

```
=== POST /resources/json/delphix/jetstream/container/ JS_DATA_CONTAINER-4
/refresh


curl -X POST -k --data
@http://172.16.160.177/resources/json/delphix/jetstream/container/
JS_DATA_CONTAINER-4 /refresh \ -b cookies.txt -H "Content-Type:
application/json" <<EOF
{}
EOF


=== RESPONSE ===
{
 "type": "OKResult",
 "status": "OK",
 "result": "",
 "job": "JOB-514",
 "action": "ACTION-924"
}
=== END ===
```

Filename: jetstream_refresh.sh or *jetstream_refresh_jq.sh*

Edit the file to update the parameters as required for your environment.

Required for Delphix Self-Service Refresh ...

```
CONTAINER_NAME="jsdc"  # Jetstream Container Name
```

Sample Output

```
$ ./jetstream_refresh.sh # or ./jetstream_refresh_jq.sh
Authenticating on http://172.16.160.195/resources/json/delphix
Session and Login Successful ...
Getting Jetstream Container Reference Value ...
container reference: JS_DATA_CONTAINER-4
abitterman-mbpro:JetStream abitterman$ vi jetstream_refresh.sh
abitterman-mbpro:JetStream abitterman$ ./jetstream_refresh.sh
Authenticating on http://172.16.160.195/resources/json/delphix
Session and Login Successful ...
Getting Jetstream Container Reference Value ...
container reference: JS_DATA_CONTAINER-4
JetStream Refresh API Results:
{"type":"OKResult","status":"OK","result":"","job":"JOB-257","action":"ACT
ION-602"}
Job: JOB-257
Current status as of Wed Aug 17 05:13:15 EDT 2016 : RUNNING 2.0% Completed
Current status as of Wed Aug 17 05:13:15 EDT 2016 : RUNNING 2.0% Completed
Current status as of Wed Aug 17 05:13:25 EDT 2016 : RUNNING 5.0% Completed
Current status as of Wed Aug 17 05:13:35 EDT 2016 : RUNNING 5.0% Completed
Current status as of Wed Aug 17 05:13:45 EDT 2016 : RUNNING 30.0% Completed
Current status as of Wed Aug 17 05:13:55 EDT 2016 : RUNNING 42.0% Completed
Current status as of Wed Aug 17 05:14:05 EDT 2016 : RUNNING 55.0% Completed
Current status as of Wed Aug 17 05:14:15 EDT 2016 : RUNNING 58.0% Completed
Current status as of Wed Aug 17 05:14:25 EDT 2016 : RUNNING 58.0% Completed
Current status as of Wed Aug 17 05:14:35 EDT 2016 : RUNNING 58.0% Completed
Current status as of Wed Aug 17 05:14:45 EDT 2016 : RUNNING 60.0% Completed
Current status as of Wed Aug 17 05:14:55 EDT 2016 : RUNNING 62.0% Completed
Current status as of Wed Aug 17 05:15:05 EDT 2016 : RUNNING 77.0% Completed
Current status as of Wed Aug 17 05:15:15 EDT 2016 : RUNNING 77.0% Completed
Job: JOB-257 COMPLETED 100.0% Completed ...


Done ...
```

Masking Use Cases

- Masking API Client
- Masking Password Change
- Masking HTTPS
- Masking in Parallel

Masking API Client

The Delphix Masking Engine now features an interactive API client that can generate commands specific to your masking engine. With those commands, you can:

- make changes to your engine
- copy and paste the commands to write code that can automate your masking activities

> **The API client will make real changes to your virtual machine. Any operations you run using the API Client will persist on the machine!**

To access the Masking API client, use the following URL: http://myMaskingEngine.com:8282/masking/api-client/, replacing "myMaskingEngine.com" with the hostname or IP address of your virtual machine.

For detailed examples of using API calls to automate masking, see the Masking API Cookbook.

Masking Password Change

https://docs.delphix.com/display/SUPPORT/How+to+create+an+encrypted+password

Masking HTTPS

https://docs.delphix.com/display/SUPPORT/Enabling+SSL+on+Masking+Engine

Masking in Parallel

Delphix masking supports launching masking jobs in parallel. When jobs have no dependies, you can initiate parallel masking API jobs (with wrapper code as required) to allow the jobs to be run as a pre and/or post hook.

Oracle Use Cases for APIs

Oracle Link + Snapshot (Sync)

The following script ingests links an environment database dSource (Oracle SID / Instance) and then takes a snapshot. See parameters for required values that you must provide.

This script demonstrates how to use name values inputs and get the respective Delphix object and/or object reference for use in the json input in downstream API calls.

Filename: link_oracle.sh # or link_oracle_jq.sh

Edit the file to update the parameters as required for your environment.

```
#########################################################
#Parameter Initialization

DMIP=172.16.160.195
#DMPORT=8282
DMUSER=delphix_admin
DMPASS=delphix
. . .


Required for Database Link and Sync ...

SOURCE_SID="DPXDEV01"   # Source Environment Database SID
SOURCE_NAME="DPXDEV01"   # Delphix dSource Name
SOURCE_ENV="Oracle Target"  # Source Environment Name
SOURCE_GRP="Oracle_Source"  # Delphix Group Name
DB_USER="delphixdb"    # Source Database SID user account
DB_PASS="delphixdb"    # Source Database SID user password

#############################################
# NO CHANGES REQUIRED BELOW THIS POINT #
#############################################


$ ./link_oracle.sh  # or ./link_oracle_jq.sh
Authenticating on http://172.16.160.195/resources/json/delphix
Session and Login Successful ...
group reference: GROUP-35
sourceconfig reference: ORACLE_SINGLE_CONFIG-1
primaryUser reference: HOST_USER-3
Linking Source Database ...
Job: JOB-92
Job: JOB-92 100.0% Completed ...
Container: ORACLE_DB_CONTAINER-19
Running SnapSync ...
Job: JOB-93
Current status as of Mon Aug 15 13:07:53 EDT 2016 : RUNNING : 0.0%
Completed
Current status as of Mon Aug 15 13:08:03 EDT 2016 : RUNNING : 15.0%
Completed
Current status as of Mon Aug 15 13:08:13 EDT 2016 : RUNNING : 35.0%
Completed
Current status as of Mon Aug 15 13:08:24 EDT 2016 : RUNNING : 59.0%
Completed
```

```
Current status as of Mon Aug 15 13:08:34 EDT 2016 : RUNNING : 66.0%
Completed
Current status as of Mon Aug 15 13:08:44 EDT 2016 : RUNNING : 74.0%
Completed
Job: JOB-93 100.0% Completed ...
Done ...
$
```

Oracle Provision

Filename: provision_oracle.txt

Shown below is how to use the CLI to provision an Oracle 11g database that is already ingested into the Delphix Engine.

The key is to get the object reference names first. For example, to get the source database container name:

```
ssh delphix_admin[delphix_engine_ip_address_or_hostname]
> database
> ls
...
> select "[database_name]"
> ls
Delphix5002HWv7 database> select 'DPXDEV01'

Delphix5002HWv7 database 'DPXDEV01'> ls
Properties
 type: OracleDatabaseContainer
 name: DPXDEV01
 . . .
  reference: ORACLE_DB_CONTAINER-18
 . . .
```

Minimum parameters required to provision:

```
Delphix5002HWv7 database provision > *commit
=== POST /resources/json/delphix/database/provision ===
{
 "type": "OracleProvisionParameters",
 "container": {
  "type": "OracleDatabaseContainer",
  "name": "VBITT" , # Delphix Object Name, Typically matches VDB name
  "group": "GROUP-36" # group ls select "[group_name]" ls
 },
 "source": {
  "type": "OracleVirtualSource",
  "mountBase": "/mnt/provision" # Delphix Filesystem Mount path
 },
 "sourceConfig": {
  "type": "OracleSIConfig",
  "repository": "ORACLE_INSTALL-3" , # repository, select
"[repository_name]"
  "databaseName": "VBITT" , # New VDB Name
  "uniqueName": "VBITT",
  "instance": {
   "type": "OracleInstance",
   "instanceName": "VBITT",
   "instanceNumber": 1
  }
 },
 "timeflowPointParameters": {
  "type": "TimeflowPointSemantic",
  "container": "ORACLE_DB_CONTAINER-18" # select "[datbase_name]" ls
 }
}
=== RESPONSE ===
```

Sample CLI Session

```
Delphix5002HWv7 database> setopt trace=false
Delphix5002HWv7 database> provision
Delphix5002HWv7 database provision > *ls
Properties
 type: OracleProvisionParameters
 container:
  type: OracleDatabaseContainer
  name: (required)
  description: (unset)
  diagnoseNoLoggingFaults: true
  group: (required)
  performanceMode: DISABLED
  preProvisioningEnabled: false
  sourcingPolicy: (unset)
 credential: (unset)
```

```
 maskingJob: (unset)
 newDBID: false
 openResetlogs: true
 physicalStandby: false
 source:
  type: OracleLiveSource
  name: (unset)
  archivelogMode: true
  config: (unset)
  configParams: (unset)
  configTemplate: (unset)
  customEnvVars: (unset)
  dataAgeWarningThreshold: 900sec
  fileMappingRules: (unset)
  manualProvisioning: false
  mountBase: (required)
  nodeListenerList: (unset)
  operations: (unset)
  redoLogGroups: 3
  redoLogSizeInMB: 0
 sourceConfig:
  type: OraclePDBConfig
  cdbConfig: (required)
  databaseName: (required)
  environmentUser: (unset)
  linkingEnabled: true
  repository: (unset)
  services: (unset)
 timeflowPointParameters:
  type: TimeflowPointSemantic
  container: (required)
  location: LATEST_POINT
 username: (unset)
Operations
defaults
Delphix5002HWv7 database provision > *edit container
Delphix5002HWv7 database provision container> *ls
Properties
 type: OracleDatabaseContainer
 name: (required)
 description: (unset)
 diagnoseNoLoggingFaults: true
 group: (required)
 performanceMode: DISABLED
 preProvisioningEnabled: false
 sourcingPolicy: (unset)
Delphix5002HWv7 database provision container> *set name=VBITT
Delphix5002HWv7 database provision container> *set group=GROUP-36
Delphix5002HWv7 database provision container> *back
Delphix5002HWv7 database provision > *edit source
Delphix5002HWv7 database provision source > *ls
Properties
 type: OracleLiveSource
```

```
 name: (unset)
 archivelogMode: true
 config: (unset)
 configParams: (unset)
 configTemplate: (unset)
 customEnvVars: (unset)
 dataAgeWarningThreshold: 900sec
 fileMappingRules: (unset)
 manualProvisioning: false
 mountBase: (required)
 nodeListenerList: (unset)
 operations: (unset)
 redoLogGroups: 3
 redoLogSizeInMB: 0
Delphix5002HWv7 database provision source > *set type=OracleVirtualSource
Delphix5002HWv7 database provision source > *set mountBase=/mnt/provision
Delphix5002HWv7 database provision source > *back
Delphix5002HWv7 database provision > *edit sourceConfig
Delphix5002HWv7 database provision sourceConfig > *ls
Properties
 type: OraclePDBConfig
 cdbConfig: (required)
 databaseName: (required)
 environmentUser: (unset)
 linkingEnabled: true
 repository: (unset)
 services: (unset)
Delphix5002HWv7 database provision sourceConfig > *set type=OracleSIConfig
Delphix5002HWv7 database provision sourceConfig > *ls
Properties
 type: OracleSIConfig
 databaseName: (required)
 environmentUser: (unset)
 instance: (required)
 linkingEnabled: true
 nonSysCredentials: (unset)
 nonSysUser: (unset)
 repository: (required)
 services: (unset)
 uniqueName: (required)
Delphix5002HWv7 database provision sourceConfig > *set databaseName=VBITT
Delphix5002HWv7 database provision sourceConfig > *set
repository=ORACLE_INSTALL-3
Delphix5002HWv7 database provision sourceConfig > *set uniqueName=VBITT
Delphix5002HWv7 database provision sourceConfig > *set
instance.instanceName=VBITT
Delphix5002HWv7 database provision sourceConfig > *set
instance.instanceNumber=1
Delphix5002HWv7 database provision sourceConfig > *ls
Properties
 type: OracleSIConfig
 databaseName: VBITT
 environmentUser: (unset)
```

```
 instance:
  type: OracleInstance
  instanceName: VBITT
  instanceNumber: 1
 linkingEnabled: true
 nonSysCredentials: (unset)
 nonSysUser: (unset)
 repository: '/u02/ora/app/product/11.2.0/dbhome_1'
 services: (unset)
 uniqueName: VBITT
Delphix5002HWv7 database provision sourceConfig > *back
Delphix5002HWv7 database provision > *edit timeflowPointParameters
Delphix5002HWv7 database provision timeflowPointParameters> *ls
Properties
 type: TimeflowPointSemantic
 container: (required)
 location: LATEST_POINT
Delphix5002HWv7 database provision timeflowPointParameters> *set
container=ORACLE_DB_CONTAINER-18
Delphix5002HWv7 database provision timeflowPointParameters> *back
Delphix5002HWv7 database provision > *commit
 VBITT
 Dispatched job JOB-348
 DB_PROVISION job started for "Oracle Target Virtual Databases/VBITT".
 Starting provision of the virtual database "VBITT".
 Creating new TimeFlow.
 Generating recovery scripts.
 Exporting storage.
 Mounting filesystems for the virtual database instance "1".
 Mounting read-only archive log filesystem for the virtual database
instance "1".
 Recovering Oracle database.
 \|/-
 Opening the virtual database "VBITT".
 Opening Oracle database.
 Oracle recovery was successful.
 Unmounting read-only archive log filesystem for the virtual database
instance "1".
 The virtual database "VBITT" was successfully provisioned.
```

```
 DB_PROVISION job for "Oracle Target Virtual Databases/VBITT" completed
successfully.
Delphix5002HWv7 database>
```

With the **setopt trace=true** option set, you can convert the JSON output from the above CLI provision command to the RESTful API cURL commands. If VBITT exists, be sure to delete it first.

Request:

```
curl X POST -k --data
@http://172.16.160.177/resources/json/delphix/database/provision \
 -b cookies.txt -H "Content-Type: application/json" <<EOF
{
 "type": "OracleProvisionParameters",
 "container": {
  "type": "OracleDatabaseContainer",
  "name": "VBITT",
  "group": "GROUP-36"
 },
 "source": {
  "type": "OracleVirtualSource",
  "mountBase": "/mnt/provision"
 },
 "sourceConfig": {
  "type": "OracleSIConfig",
  "repository": "ORACLE_INSTALL-3",
  "databaseName": "VBITT",
  "uniqueName": "VBITT",
  "instance": {
   "type": "OracleInstance",
   "instanceName": "VBITT",
   "instanceNumber": 1
  }
 },
 "timeflowPointParameters": {
  "type": "TimeflowPointSemantic",
  "container": "ORACLE_DB_CONTAINER-18"
 }
}
EOF
```

Response:

```
{"type":"OKResult","status":"OK","result":"ORACLE_DB_CONTAINER-22","job":"
JOB-353","action":"ACTION-649"}
```

Put all the commands above within a shell script to automate the complete process of provisioning an Oracle 11.2.0.4 database.

Notice that the script below looks up 4 object references for use within the JSON input into the API.

Filename: provision_oracle.sh# or provision_oracle_jq.sh

Edit the file to update the parameters as required for your environment.

```
######################
# DELPHIX CORP #
######################
#Parameter Initialization

DMIP=172.16.160.195
DMUSER=delphix_admin
DMPASS=delphix
COOKIE="~/cookies.txt"
COOKIE=`eval echo $COOKIE`
CONTENT_TYPE="Content-Type: application/json"
DELAYTIMESEC=10
BaseURL="http://${DMIP}/resources/json/delphix"
#
Required for Database Link and Sync ...
#
VDB_NAME="VBITT"    # Delphix VDB Name
MOUNT_BASE="/mnt/provision"  # Delphix Engine Mount Path
SOURCE_GRP="Oracle_Target"  # Delphix Engine Group Name
TARGET_ENV="Oracle Target"  # Target Environment used to get repository
reference value
SOURCE_SID="DPXDEV01"   # dSource name used to get db container reference
value
################################################
# NO CHANGES REQUIRED BELOW THIS POINT #
################################################
```

Sample Output

```
$ ./provision_oracle.sh# or ./provision_oracle_jq.sh
Authenticating on http://172.16.160.195/resources/json/delphix
Session and Login Successful ...
group reference: GROUP-36
container reference: ORACLE_DB_CONTAINER-36
env reference: UNIX_HOST_ENVIRONMENT-3
repository reference: ORACLE_INSTALL-1
Provisioning VDB from Source Database ...
Job: JOB-155
Current status as of Mon Aug 15 23:40:51 EDT 2016 : RUNNING 0.0% Completed
Current status as of Mon Aug 15 23:40:51 EDT 2016 : RUNNING 0.0% Completed
Current status as of Mon Aug 15 23:41:01 EDT 2016 : RUNNING 9.0% Completed
Current status as of Mon Aug 15 23:41:11 EDT 2016 : RUNNING 45.0% Completed
Current status as of Mon Aug 15 23:41:21 EDT 2016 : RUNNING 45.0% Completed
Current status as of Mon Aug 15 23:41:31 EDT 2016 : RUNNING 46.0% Completed
Current status as of Mon Aug 15 23:41:41 EDT 2016 : RUNNING 48.0% Completed
Current status as of Mon Aug 15 23:41:51 EDT 2016 : RUNNING 60.0% Completed
Job: JOB-155 COMPLETED 100.0% Completed ...
Done ...
$
```

Filename: provision_oracle_child.sh# or provision_oracle_child_jq.sh

```
$ ./provision_oracle_child.sh# or ./provision_oracle_child_jq.sh
Authenticating on http://172.16.160.195/resources/json/delphix
Session and Login Successful ...
group reference: GROUP-36
container reference: ORACLE_DB_CONTAINER-118
env reference: UNIX_HOST_ENVIRONMENT-9
repository reference: ORACLE_INSTALL-6
Provisioning VDB from Source Database ...
Job: JOB-857
Current status as of Mon Sep 5 22:48:28 EDT 2016 : RUNNING 0.0% Completed
Current status as of Mon Sep 5 22:48:28 EDT 2016 : RUNNING 0.0% Completed
Current status as of Mon Sep 5 22:48:38 EDT 2016 : RUNNING 9.0% Completed
Current status as of Mon Sep 5 22:48:48 EDT 2016 : RUNNING 27.0% Completed
Current status as of Mon Sep 5 22:48:58 EDT 2016 : RUNNING 42.0% Completed
Current status as of Mon Sep 5 22:49:08 EDT 2016 : RUNNING 45.0% Completed
Current status as of Mon Sep 5 22:49:28 EDT 2016 : RUNNING 46.0% Completed
Current status as of Mon Sep 5 22:49:38 EDT 2016 : RUNNING 48.0% Completed
Current status as of Mon Sep 5 22:49:48 EDT 2016 : RUNNING 51.0% Completed
Current status as of Mon Sep 5 22:50:08 EDT 2016 : RUNNING 71.0% Completed
Job: JOB-857 COMPLETED 100.0% Completed ...
Done ...
```

SQL Server API Use Cases

- SQL Server Link/Ingest Environment dSource
- SQL Server Provision

- SQL Server Refresh

SQL Server Link/Ingest Environment dSource

For the **Window Target** environment, the dSource **delphixdb** in **MSSQLSERVER** instance will be linked/ingested into the Delphix Engine. It will appear in the **Windows_Source** group below.

Filename: link_sqlserver.ps1

```
PS> . .\link_sqlserver.ps1
Authenticating on http://172.16.160.195/resources/json/delphix
Login Successful ...
group reference: GROUP-34
sourceconfig reference: MSSQL_SINGLE_CONFIG-26
env reference: WINDOWS_ HOST_ENVIRONMENT-7
repository reference: MSSQL_INSTANCE-4
database link API Results:
{"type":"OKResult","status":"OK","result":"MSSQL_DB_CONTAINER-114","job":"
JOB-819","action":
"ACTION-1659"}
DB Container: MSSQL_DB_CONTAINER-114
```

```
Job # JOB-819
***** waiting for status *****
Current status as of 09/05/2016 11:41:13 : COMPLETED : 100.0% Completed
Job COMPLETED Succesfully.
```

```
JOB JOB-820
waiting for status *****
Current status as of 09/05/2016 11:41:23 : RUNNING : 5.0% Completed
Current status as of 09/05/2016 11:41:44 : RUNNING : 9.0% Completed
```

```
Current status as of 09/05/2016 11:41:54 : RUNNING : 56.0% Completed
Job COMPLETED Succesfully.
Done ...
```

Successful dSource linked/ingested into the Delphix Engine.

SQL Server Provision

The example below is done from the command line once you know the parameters and reference object names.

Filename: windows_sqlserver_provision.txt

Create these 3 JSON text files:

```
session.json
```

```
{
 "type": "APISession",
 "version": {
  "type": "APIVersion",
  "major": 1,
  "minor": 5,
  "micro": 3
 }
}
login.json
{
 "type": "LoginRequest",
 "username": "delphix_admin",
 "password": "delphix"
}
provision.json
{
 "type": "MSSqlProvisionParameters",
 "container": {
  "type": "MSSqlDatabaseContainer",
  "name": "Vbitt00",
  "group": "GROUP-36",
  "sourcingPolicy": {
   "type": "SourcingPolicy",
   "loadFromBackup": false,
   "logsyncEnabled": false
  },
  "validatedSyncMode": "TRANSACTION_LOG"
 },
 "source": {
  "type": "MSSqlVirtualSource",
  "operations": {
   "type": "VirtualSourceOperations",
   "configureClone": [],
   "postRefresh": [],
   "postRollback": [],
   "postSnapshot": [],
   "preRefresh": [],
   "preSnapshot": []
  }
 },
 "sourceConfig": {
  "type": "MSSqlSIConfig",
  "linkingEnabled": false,
  "repository": "MSSQL_INSTANCE-1",
  "databaseName": "Vbitt00",
  "recoveryModel": "SIMPLE",
  "instance": {
   "type": "MSSqlInstanceConfig",
   "host": "WINDOWS_HOST-1"
  }
 },
 "timeflowPointParameters": {
```

```
    "type": "TimeflowPointSemantic",
    "container": "MSSQL_DB_CONTAINER-23",
    "location": "LATEST_SNAPSHOT"
  }
}
```

This works on Windows Powershell Command Prompt

> Use curl, curl.exe or modify the default alias.

```
curl --insecure -c cookies.txt -i -X POST -H "Content-Type:
application/json" -d "@session.json"
http://172.16.160.153/resources/json/delphix/session

curl --insecure -b cookies.txt -i -X POST -H "Content-Type:
application/json" -d "@login.json"
http://172.16.160.153/resources/json/delphix/login

curl --insecure -b cookies.txt -i -X POST -H "Content-Type:
application/json" -d "@provision.json"
http://172.16.160.153/resources/json/delphix/database/provision
```

Plug in the returned JOB #

```
curl --insecure -b cookies.txt -i -X GET -H "Content-Type:
application/json" -k
http://172.16.160.153/resources/json/delphix/notification?channel=JOB-428
```

Get Example

```
curl --insecure -b cookies.txt -i -X GET -H "Content-Type:
application/json" -k http://172.16.160.153/resources/json/delphix/system
```

Complete example.

Provision the newly created **delphixdb** dSource in the **Windows_Source** group to a virtual database VBITT in the **Windows_Target** group.

Filename: provision_sqlserver.ps1

Variables ...

```
$nl = [Environment]::NewLine
$BaseURL = " http://172.16.160.195/resources/json/delphix "
$cookie = "cookies.txt"
$delphix_user = "delphix_admin"
$delphix_pass = "delphix"
. . .
```

Required for Provisioning Virtual Database ...

```
$SOURCE_SID="delphixdb"   # dSource name used to get db container reference
value


$VDB_NAME="VBITT"    # Delphix VDB Name
$TARGET_GRP="Windows_Target" # Delphix Engine Group Name
$TARGET_ENV="Window Target"  # Target Environment used to get repository
reference value
$TARGET_REP="MSSQLSERVER" # Target Environment Repository / Instance name

##############################################
# NO CHANGES REQUIRED BELOW THIS POINT #
##############################################
```

Sample Run Output

```
PS> . .\provision_sqlserver.ps1
Authenticating on http://172.16.160.195/resources/json/delphix
Login Successful ...
group reference:  GROUP-37
container reference:  MSSQL_DB_CONTAINER-114
env reference:  WINDOWS_HOST_ENVIRONMENT-7
repository reference:  MSSQL_INSTANCE-4
database provision API Results:
{"type":"OKResult","status":"OK","result":"MSSQL_DB_CONTAINER-115","job":"
JOB-822","act
ion":"ACTION-1664"}
DB Container:  MSSQL_DB_CONTAINER-115

Job # JOB-822


jobState RUNNING
percentComplete 0.0
***** waiting for status *****
Current status as of 09/05/2016 11:43:51 : RUNNING : 0.0% Completed
Current status as of 09/05/2016 11:44:01 : RUNNING : 3.0% Completed
Current status as of 09/05/2016 11:44:12 : RUNNING : 11.0% Completed
Current status as of 09/05/2016 11:44:22 : RUNNING : 18.0% Completed
Current status as of 09/05/2016 11:44:32 : RUNNING : 18.0% Completed
```

```
Current status as of 09/05/2016 11:44:52 : RUNNING : 75.0% Completed
Job COMPLETED Succesfully.

Done ...
```

SQL Server Refresh

The following are curl commands that can be issued from the Powershell command line. For inclusion within a Powershell script, see the masking example, *masking.ps1*.

Filename: windows_sqlserver_refresh.txt

MS SQL Server Refresh Example ...

Session ...

```
curl --insecure -c cookies.txt -i -X POST -H "Content-Type:
application/json" -d "@session.json"
http://172.16.160.179/resources/json/delphix/session
```

Filename: session.json

```
{
 "type": "APISession",
 "version": {
  "type": "APIVersion",
  "major": 1,
  "minor": 5,
  "micro": 3
 }
}

PS> *curl --insecure -c cookies.txt -i -X POST -H "Content-Type:
application/json" -d "@session.json"
http://172.16.160.179/resources/json/delphix/session*
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Set-Cookie: JSESSIONID=8DE0362F5BBD73E6BFA9E13FF111E78C; Path=/resources/;
HttpOnly
Content-Type: application/json
Content-Length: 179
Date: Thu, 16 Jun 2016 07:24:34 GMT

{"type":"OKResult","status":"OK","result":{"type":"APISession","version":{
"type":"APIVersion","major":1,"minor":5,"micro":3},"locale":null,"client":
null},"job":null,"action":null}
PS>
```

Login ...

```
curl --insecure -b cookies.txt -i -X POST -H "Content-Type:
application/json" -d "@login.json"
http://172.16.160.179/resources/json/delphix/login
```

Filename:login.json

```
{
 "type": "LoginRequest",
 "username": "delphix_admin",
 "password": "delphix"
}

PS> *curl --insecure -b cookies.txt -i -X POST -H "Content-Type:
application/json" -d "@login.json"
http://172.16.160.179/resources/json/delphix/login*
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/json
Content-Length: 76
Date: Thu, 16 Jun 2016 07:25:39 GMT


{"type":"OKResult","status":"OK","result":"USER-2","job":null,"action":nul
l}
PS C:\Users\Administrator>
```

List Databases ...

```
curl --insecure -b cookies.txt -i -X GET -H "Content-Type:
application/json" -k http://172.16.160.179/resources/json/delphix/database


PS> *curl --insecure -b cookies.txt -i -X GET -H "Content-Type:
application/json" -k http://172.16.160.179/resources/json/delphix/database*
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/json
Content-Length: 4062
Date: Thu, 16 Jun 2016 07:27:14 GMT

{"type":"ListResult","status":"OK","result":[...
...
{"type":"MSSqlDatabaseContainer","reference":"MSSQL_DB_CONTAINER-37","name
space":null,"name":"Vdelphix_demo","group":"GR
OUP-35","provisionContainer":"MSSQL_DB_CONTAINER-36","creationTime":"2016-
06-16T07:09:06.222Z","currentTimeflow":"MSSQL_
TIMEFLOW-38","previousTimeflow":"MSSQL_TIMEFLOW-37","description":null,"ru
ntime":
...
{"type":"So{"type":"MSSqlDatabaseContainer","reference":"MSSQL_DB_CONTAINE
R-36","namespace":null,"name":"delphix_demo","
group":"GROUP-35","provisionContainer":null,"creationTime":"2016-06-16T07:
07:49.939Z","currentTimeflow":"MSSQL_TIMEFLOW-
36","previousTimeflow":null,"description":"","runtime":
...
...}],"job":null,"action":null,"total":6,"overflow":false}


PS>
```

Need Reference Object from Database Information ...

For Parent Source Database delphix_demo, reference object is MSSQL_DB_CONTAINER-36

For Virtual Database Vdelphix_demo, reference object is MSSQL_DB_CONTAINER-37

[Optional: Get Database Info ...]

```
curl --insecure -b cookies.txt -i -X GET -H "Content-Type:
application/json" -k
http://172.16.160.179/resources/json/delphix/database/MSSQL_DB_CONTAINER-37

PS> *curl --insecure -b cookies.txt -i -X GET -H "Content-Type:
application/json" -k
http://172.16.160.179/resources/json/delphix/database/MSSQL_DB_CONTAINER-3
7*
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/json
Content-Length: 696
Date: Thu, 16 Jun 2016 07:35:42 GMT

{"type":"OKResult","status":"OK","result":{"type":"MSSqlDatabaseContainer"
,"reference":"MSSQL_DB_CONTAINER-37","namespac
e":null,"name":"Vdelphix_demo","group":"GROUP-35","provisionContainer":"MS
SQL_DB_CONTAINER-36","creationTime":"2016-06-1
6T07:09:06.222Z","currentTimeflow":"MSSQL_TIMEFLOW-38","previousTimeflow":
"MSSQL_TIMEFLOW-37","description":null,"runtim
e":{"type":"MSSqlDBContainerRuntime","logSyncActive":false,"preProvisionin
gStatus":null,"lastRestoredBackupSetUUID":null
},"os":"Windows","processor":"x86","sourcingPolicy":{"type":"SourcingPolic
y","logsyncEnabled":false,"loadFromBackup":fal
se},"performanceMode":"DISABLED","delphixManaged":true,"masked":false},"jo
b":null,"action":null}


PS>
```

Refresh Vdelphix_demo using parent delphix_demo (MSSQL_DB_CONTAINER-36) with the latest timecard ...

```
curl --insecure -b cookies.txt -i -X POST -H "Content-Type:
application/json" -d "@refresh.json"
http://172.16.160.179/resources/json/delphix/database/MSSQL_DB_CONTAINER-3
7/refresh


=== POST /resources/json/delphix/database/MSSQL_DB_CONTAINER-37/refresh ===
refresh.json
{
 "type": "RefreshParameters",
 "timeflowPointParameters": {
  "type": "TimeflowPointSemantic",
  "container": "MSSQL_DB_CONTAINER-36"
 }
}
PS> *curl --insecure -b cookies.txt -i -X POST -H "Content-Type:
application/json" -d "@refresh.json"
http://172.16.160.179/resources/json/delphix/database/MSSQL_DB_CONTAINER-3
7/refresh*
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/json
Content-Length: 82
Date: Thu, 16 Jun 2016 07:40:44 GMT


{"type":"OKResult","status":"OK","result":"","job":"JOB-60","action":"ACTI
ON-167"}


PS>
```

[Observer Delphix GUI Action]

Done with SQL Server VDB Refresh ...

API Programming Language Examples

The following programming language examples are just to show the bare minimum authentication and a sample functional API call. There are numerous modules, libraries, methods, functions, and code examples to further demonstrate how the languages work with the Delphix APIs and JSON data strings/objects.

You can execute PHP, Perl, and Python languages from the command line and/or from within a Web Server such as Apache or IIS. The following examples are formatted for command line / terminal output.

PHP

PHP provides cURL and JSON modules.

```
$ php -i | grep -iE "cURL|json"
curl
cURL support => enabled
cURL Information => 7.43.0
json
json support => enabled
json version => 1.2.1
```

Filename: delphix_curl.php

Sample Output:

```
$ php -f delphix_curl.php
Session json>
{"type":"APISession","version":{"type":"APIVersion","major":1,"minor":7,"m
icro":0}}
Session Results>
{"type":"OKResult","status":"OK","result":{"type":"APISession","version":{
"type":"APIVersion","major":1,"minor":7,"micro":0},"locale":null,"client":
null},"job":null,"action":null}
Login json>
{"type":"LoginRequest","username":"delphix_admin","password":"delphix"}
Login Results>
{"type":"OKResult","status":"OK","result":"USER-2","job":null,"action":nul
l}
Calling About API ...
About Results>
{"type":"OKResult","status":"OK","result":{"type":"PublicSystemInfo","prod
uctType":"standard","productName":"Delphix Engine","buildTitle":"Delphix
Engine
5.1.1.0","buildTimestamp":"2016-07-21T07:23:41.000Z","buildVersion":{"type
":"VersionInfo","major":5,"minor":1,"micro":1,"patch":0},"configured":true
,"enabledFeatures":["XPP","MSSQLHOOKS"],"apiVersion":{"type":"APIVersion",
"major":1,"minor":8,"micro":0},"banner":null,"locales":["en-US"],"currentL
ocale":"en-US"},"job":null,"action":null}
Converting json string to a PHP Array
stdClass Object
(
 [type] => OKResult
 [status] => OK
 [result] => stdClass Object
  (
   [type] => PublicSystemInfo
   [productType] => standard
   [productName] => Delphix Engine
   [buildTitle] => Delphix Engine 5.1.1.0
   [buildTimestamp] => 2016-07-21T07:23:41.000Z
   [buildVersion] => stdClass Object
    (
     [type] => VersionInfo
```

```
        [major] => 5
        [minor] => 1
        [micro] => 1
        [patch] => 0
       )
      [configured] => 1
      [enabledFeatures] => Array
       (
        [0] => XPP
        [1] => MSSQLHOOKS
       )

      [apiVersion] => stdClass Object
       (
        [type] => APIVersion
        [major] => 1
        [minor] => 8
        [micro] => 0
       )
      [banner] =>
      [locales] => Array
       (
        [0] => en-US
       )
      [currentLocale] => en-US
     )
```

```
 [job] =>
 [action] =>
)
```

Perl

Perl provides a couple of methods for working with cURL: operating system calls, WWW::Curl (libcurl) module, or LWP::Curl module. The sample below simply logs into the Delphix Engine and lists the current Delphix Environments.

Filename: perl_curl.pl

Sample Output:

```
$ perl perl_curl.pl
Testing cURL on Perl ...

Session Results:
{"type":"OKResult","status":"OK","result":{"type":"APISession","version":{
"type":"APIVersion","major":1,"minor":7,"micro":0},"locale":null,"client":
null},"job":null,"action":null}
Login Results:
{"type":"OKResult","status":"OK","result":"USER-2","job":null,"action":nul
l}


Enviornment Results:
{"type":"ListResult","status":"OK","result":[{"type":"WindowsHostEnvironme
nt","reference":"WINDOWS_HOST_ENVIRONMENT-7","namespace":null,"name":"Wind
ow
Target","description":null,"primaryUser":"HOST_USER-7","enabled":false,"ho
st":"WINDOWS_HOST-6","proxy":null},{"type":"UnixHostEnvironment","referenc
e":"UNIX_HOST_ENVIRONMENT-9","namespace":null,"name":"Oracle
Target","description":"","primaryUser":"HOST_USER-9","enabled":true,"host"
:"UNIX_HOST-8","aseHostEnvironmentParameters":null}],"job":null,"action":n
ull,"total":2,"overflow":false}


Done
```

Python

Delphix has an extensive resource library for using Python with the Delphix Engine.

https://docs.delphix.com/display/DOCS/CLI+to+Python+Transition

Delphix python module

Blogs

https://github.com/CloudSurgeon/delphixpy-examples

Related Videos

- https://vimeo.com/164779308
- https://vimeo.com/170187276
- https://vimeo.com/170896907

Simple Python program to authenticate and get the "about" API results. This script requires the "request" and "json" modules.

http://stackoverflow.com/questions/17309288/importerror-no-module-named-requests

Filename: auth.py

Sample Output

```
$ python auth.py
Authenticating URL http://172.16.160.195/resources/json/delphix ...
{"type":"OKResult","status":"OK","result":{"type":"APISession","version":{
"type":"APIVersion","major":1,"minor":7,"micro":0},"locale":null,"client":
null},"job":null,"action":null}
Login ...
{"type":"OKResult","status":"OK","result":"USER-2","job":null,"action":nul
l}
About ...
{"type":"OKResult","status":"OK","result":{"type":"PublicSystemInfo","prod
uctType":"standard","productName":"Delphix Engine","buildTitle":"Delphix
Engine
5.1.1.0","buildTimestamp":"2016-07-21T07:23:41.000Z","buildVersion":{"type
":"VersionInfo","major":5,"minor":1,"micro":1,"patch":0},"configured":true
,"enabledFeatures":["XPP","MSSQLHOOKS"],"apiVersion":{"type":"APIVersion",
"major":1,"minor":8,"micro":0},"banner":null,"locales":["en-US"],"currentL
ocale":"en-US"},"job":null,"action":null}
JSON Parsing Examples ...
OK
Delphix Engine 5.1.1.0
1
```

JSP (Java Server Pages)

Java Server Pages are typically used for the web formatting and output, but you can also use JSP for application logic processing and native Java code integration, although this is scorned by the purest and most logical thinking programmers.

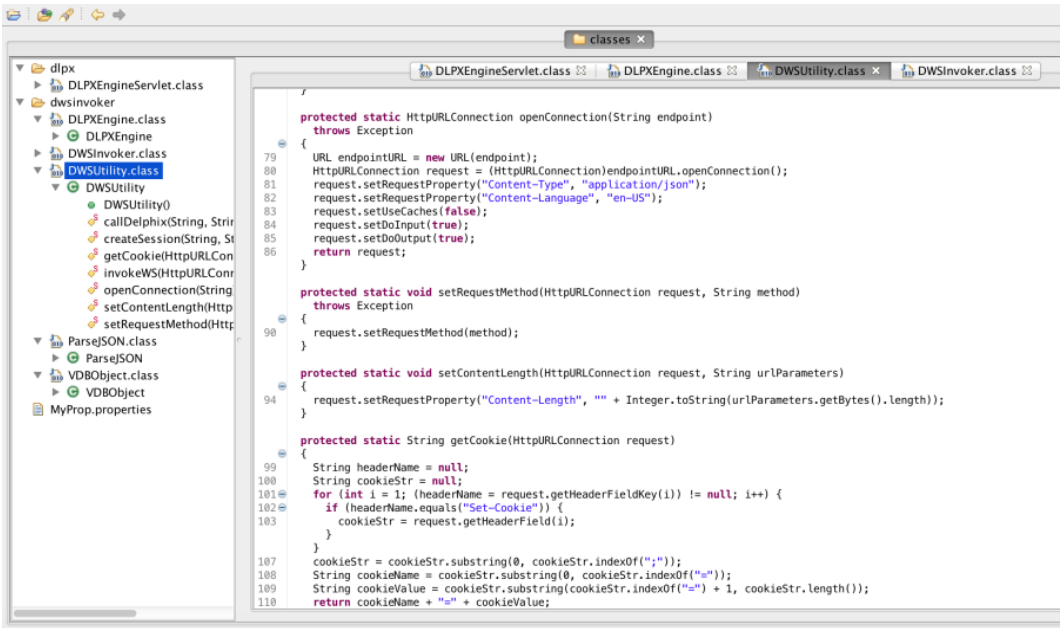
Filename: delphix_http.jsp

Sample Output:

Browser URL: **http://localhost:8080/delphix_http.jsp**

Java

Java methods and classes allow coding logic to be effectively re-used, extended and modularized for flexible applications. Using the Java code embedded within the JSP file, code is logically placed into respective classes and methods.



API Timeflows

From earlier, the RESTFul URL for a virtual database refresh will look like:

http://<delphix_engine>/resources/json/delphix/database/MSSQL_DB_CONTAINER-39/refresh

where the **MSSQL_DB_CONTAINER-39** represents the target virtualized database to refresh and we need to POST the JSON data to the URL for processing.

```
{
 "type": "RefreshParameters",
 "timeflowPointParameters": {
  "type": "TimeflowPointSemantic",
  "container": "MSSQL_DB_CONTAINER-38"
 }
}
```

Timeflow Parameters

The **"timeflowPointParameters"** key has 6 **"type": "..."** options which each have their own set of parameters. The type **"TimeflowPointSemantic"** uses the default LATEST_POINT within the source container. Now, for more on timeflowPointParameters.

http://<delphix_engine>/api/#TimeflowPointParameters

TimeflowPointParameters

Parameters indicating a TimeFlow point to use as input to database operations.

TypedObject
TimeflowPointParameters

Direct Known Subclasses:

TimeflowPointTimestamp, TimeflowPointSnapshot, TimeflowPointSemantic, TimeflowPointLocation, TimeflowPointBookmark, TimeflowPointBookmarkTag

TimeflowPointTimestamp

timeflow	Reference to TimeFlow containing this point. Type: Reference to Timeflow Constraints: Required: true
timestamp	The logical time corresponding to the TimeFlow location. Type: date Constraints: Required: true

TimeflowPointSnapshot

snapshot	Reference to the snapshot. Type: Reference to TimeflowSnapshot

TimeflowPointSemantic

Semantic reference to a TimeFlow point.

> The reference is relative to a container and not a TimeFlow. If the container contains multiple TimeFlows, the Delphix Engine will evaluate the semantic reference with regards to all TimeFlows in that container.

container	Reference to the container. Type: Reference to Container Constraints: Required: true
location	A semantic description of a TimeFlow location. Type: string Constraints: Default: LATEST_POINT Acceptable values: LATEST_POINT, LATEST_SNAPSHOT Create: optional Update: optional

TimeflowPointLocation

TimeFlow point based on a database-specific identifier (SCN, LSN, etc).

location	The TimeFlow location. Type: string Constraints: Required: true
timeflow	Reference to TimeFlow containing this location. Type: Reference to Timeflow Constraints: Required: true

TimeflowPointBookmark

bookmark	Reference to the bookmark. Type: Reference to TimeflowBookmark Constraints: Required: true

TimeflowPointBookmarkTag

container	Reference to the container. Type: Reference to Container Constraints: Required: true
tag	The name of the tag. Type: string Constraints: Required: true

Timeflow API Objects: timeflow, snapshot, timeflowRanges

The sample code provides information on the timeflow, timeflowRanges and snapshot objects for the respective VDB.

Filename: flows.sh

```
$ ./flows.sh VBITT10
Session API

Login API
Login
{"type":"OKResult","status":"OK","result":"USER-2","job":null,"action":nul
l}
Source: VBITT10
container reference: ORACLE_DB_CONTAINER-75

Timeflows API
timeflow names:
DB_PROVISION@2016-09-27T13:15:18
DB_ROLLBACK@2016-09-28T00:33:54
DB_ROLLBACK@2016-09-28T00:38:33
DB_ROLLBACK@2016-09-28T00:41:44
```

Select timeflow Name (copy-n-paste from above list):

```
DB_ROLLBACK@2016-09-28T00:33:54
timeflow reference: ORACLE_TIMEFLOW-97
```

TimeflowRanges for this timeflow ...

```
{
 "type": "ListResult",
 "status": "OK",
 "result": [
  {
   "type": "TimeflowRange",
   "startPoint": {
    "type": "OracleTimeflowPoint",
    "location": "5475918",
    "timestamp": "2016-09-28T04:35:39.000Z",
    "timeflow": "ORACLE_TIMEFLOW-97"
   },
   "endPoint": {
    "type": "OracleTimeflowPoint",
    "location": "5476181",
    "timestamp": "2016-09-28T04:37:53.000Z",
    "timeflow": "ORACLE_TIMEFLOW-97"
   },
   "provisionable": true
  }
 ],
 "job":null,
 "action":null,
 "total": 1,
 "overflow": false
}
```


Snapshot per Timeflow ...

snapshots:
@2016-09-28T04:35:39.826Z
@2016-09-28T04:37:38.537Z
@2016-09-28T04:37:51.273Z

Select Snapshot Name (copy-n-paste from above list):

```
@2016-09-28T04:37:38.537Z
snapshot reference: ORACLE_SNAPSHOT-152
{
 "type": "OracleSnapshot",
 "reference": "ORACLE_SNAPSHOT-152",
 "namespace":null,
 "name": "@2016-09-28T04:37:38.537Z",
 "consistency": "CRASH_CONSISTENT",
 "missingNonLoggedData": false,
 "container": "ORACLE_DB_CONTAINER-75",
 "creationTime": "2016-09-28T04:37:38.537Z",
 "firstChangePoint": {
  "type": "OracleTimeflowPoint",
  "location": "5476152",
  "timestamp":null,
  "timeflow": "ORACLE_TIMEFLOW-97"
 },
 "latestChangePoint": {
  "type": "OracleTimeflowPoint",
  "location": "5476157",
  "timestamp": "2016-09-28T04:37:38.000Z",
  "timeflow": "ORACLE_TIMEFLOW-97"
 },
 "retention": 0,
 "timeflow": "ORACLE_TIMEFLOW-97",
 "timezone": "US/Eastern,EDT-0400",
 "version": "11.2.0.4.0",
 "runtime": {
  "type": "OracleSnapshotRuntime",
  "provisionable": true,
  "missingLogs":null
 },
 "temporary": false,
 "fromPhysicalStandbyVdb": false,
 "fractionTimeflows":null,
 "redoLogSizeInBytes": 52428800
}

Done
```

The following scripts demonstrate REFRESH and RESET/ROLLBACK for the timeflow types of timestamp, snapshot and scn/lsn.

Filename: vdb_refresh_timestamp.sh
Filename: vdb_refresh_snapshot.sh
Filename: vdb_refresh_scn.sh

Filename: vdb_rollback_timestamp.sh
Filename: vdb_rollback_snapshot.sh
Filename: vdb_rollback_scn.sh

Usage: ./vdb_[*all_the_above_scripts*].sh [source_vdb_name]

Sample Usage: Copy-and-Paste the timeflow name to the "Select timeflow Name:" prompt. Enter any timestamp between the startPoint and endPoint values using the format:
[yyyy] - [MM] - [dd]T[HH] : [mm] : [ss] . [SSS]Z

```
$ ./vdb_rollback_timestamp.sh VBITT
Session and Login Successful ...
database container reference: ORACLE_DB_CONTAINER-73

Timeflows API
Timeflow Names:
DB_REFRESH@2016-09-28T00:13:13
DB_PROVISION@2016-09-26T07:00:08
DB_ROLLBACK@2016-09-26T07:45:30
DB_REFRESH@2016-09-26T12:20:57
DB_ROLLBACK@2016-09-27T12:29:47
```

Select Timeflow Name (copy-n-paste from above list):

```
DB_ROLLBACK@2016-09-26T07:45:30

Timeflow Reference: ORACLE_TIMEFLOW-89
TimeflowRanges for this timeflow ...
{
 "type": "ListResult",
 "status": "OK",
 "result": [
  {
   "type": "TimeflowRange",
   "startPoint": {
    "type": "OracleTimeflowPoint",
    "location": "5474012",
    "timestamp": "2016-09-26T15:30:35.000Z",
    "timeflow": "ORACLE_TIMEFLOW-89"
   },
   "endPoint": {
    "type": "OracleTimeflowPoint",
    "location": "5482482",
    "timestamp": "2016-09-26T21:15:17.000Z",
    "timeflow": "ORACLE_TIMEFLOW-89"
   },
   "provisionable": true
  }
 ],
 "job": null,
 "action": null,
 "total": 1,
 "overflow": false
}

Timestamp Format "[yyyy][MM][dd]T[HH]:[mm]:[ss].[SSS]Z"
Enter Timestamp between Start and End Point values (exclude quotes):
2016-09-26T15:45:00.000Z

json> {
 "type": "OracleRollbackParameters",
 "timeflowPointParameters": {
  "type": "TimeflowPointTimestamp",
  "timeflow": "ORACLE_TIMEFLOW-89",
  "timestamp": "2016-09-26T15:45:00.000Z"
 },
 "username": ""
}
Job: JOB-515
Current status as of Wed Sep 28 00:49:39 EDT 2016 : RUNNING 0% Completed
. . .
Current status as of Wed Sep 28 00:51:10 EDT 2016 : RUNNING 73% Completed
Current status as of Wed Sep 28 00:51:50 EDT 2016 : RUNNING 96% Completed
Job: JOB-515 COMPLETED 100% Completed ...
Done ...
```