Veritas™ Resiliency Platform 2.2 Application Enablement SDK

VERITAS™

Veritas Resiliency Platform: Application Enablement SDK

Last updated: 2017-03-30

Document version: Document version: 2.2 Rev 0

Legal Notice

Copyright © 2017 Veritas Technologies LLC. All rights reserved.

Veritas, the Veritas Logo, Veritas InfoScale, and NetBackup are trademarks or registered trademarks of Veritas Technologies LLC or its affiliates in the U.S. and other countries. Other names may be trademarks of their respective owners.

This product may contain third party software for which Veritas is required to provide attribution to the third party (“Third Party Programs”). Some of the Third Party Programs are available under open source or free software licenses. The License Agreement accompanying the Software does not alter any rights or obligations you may have under those open source or free software licenses. Refer to the third party legal notices document accompanying this Veritas product or available at:

https://www.veritas.com/about/legal/license-agreements

The product described in this document is distributed under licenses restricting its use, copying, distribution, and decompilation/reverse engineering. No part of this document may be reproduced in any form by any means without prior written authorization of Veritas Technologies LLC and its licensors, if any.

THE DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID. VERITAS TECHNOLOGIES LLC SHALL NOT BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS DOCUMENTATION. THE INFORMATION CONTAINED IN THIS DOCUMENTATION IS SUBJECT TO CHANGE WITHOUT NOTICE.

The Licensed Software and Documentation are deemed to be commercial computer software as defined in FAR 12.212 and subject to restricted rights as defined in FAR Section 52.227-19 "Commercial Computer Software - Restricted Rights" and DFARS 227.7202, et seq. "Commercial Computer Software and Commercial Computer Software Documentation," as applicable, and any successor regulations, whether delivered by Veritas as on premises or hosted services. Any use, modification, reproduction release, performance, display or disclosure of the Licensed Software and Documentation by the U.S. Government shall be solely in accordance with the terms of this Agreement.

Veritas Technologies LLC

500 E Middlefield Road
Mountain View, CA 94043

http://www.veritas.com

Technical Support

Technical Support maintains support centers globally. All support services will be delivered in accordance with your support agreement and the then-current enterprise technical support policies. For information about our support offerings and how to contact Technical Support, visit our website:

https://www.veritas.com/support

You can manage your Veritas account information at the following URL:

https://my.veritas.com

If you have questions regarding an existing support agreement, please email the support agreement administration team for your region as follows:

Worldwide (except Japan)	CustomerCare@veritas.com
Japan	CustomerCare_Japan@veritas.com

Documentation

Make sure that you have the current version of the documentation. Each document displays the date of the last update on page 2. The document version appears on page 2 of each guide. The latest documentation is available on the Veritas website:

https://sort.veritas.com/documents

Documentation feedback

Your feedback is important to us. Suggest improvements or report errors or omissions to the documentation. Include the document title, document version, chapter title, and section title of the text on which you are reporting. Send feedback to:

doc.feedback@veritas.com

You can also see documentation information or ask a question on the Veritas community site:

http://www.veritas.com/community/

Veritas Services and Operations Readiness Tools (SORT)

Veritas Services and Operations Readiness Tools (SORT) is a website that provides information and tools to automate and simplify certain time-consuming administrative tasks. Depending on the product, SORT helps you prepare for installations and upgrades, identify risks in your datacenters, and improve operational efficiency. To see what services and tools SORT provides for your product, see the data sheet:

https://sort.veritas.com/data/support/SORT_Data_Sheet.pdf

Contents

Chapter 1

Introduction

This chapter includes the following topics:

Introduction to Application Enablement SDK

The Application Enablement Software Development Kit (SDK) lets you write application scripts to discover and manage the applications in your data center.

The application bundle is in a `.tar.gz` format that contains a script and related modules that can discover all the instances of a particular type of application in your data center for the supported platforms. The scripts contained in the file are able to start or stop a single instance of that application. The application bundle also contains a `bundle.ini` file that provides some basic information about the bundle to the Resiliency Platform.

The application bundle can be uploaded on the Resiliency Manager and then deployed on all the managed hosts in your data center. When the applications are discovered and reported on the Resiliency Manager console, you can then organize them into resiliency groups that can be protected and managed as a single entity.

For more information on how to upload the application bundle on the Resiliency Manager, install and enable on hosts, refer to *Solutions for Applications* guide.

Chapter 2

Managing Perl APIs

This chapter includes the following topics:

About Perl APIs

The Perl APIs is a simplified set of APIs which you can use to write an application module script that helps you discover and operate your applications.

The Perl APIs must include the following two modules:

- Application module
  This module provides the functionality to add, discover, or operate on an application using the Resiliency Platform web console.
- Constants module
  This module provides the functionality to fetch the constants that are required to set the log levels, or to fetch the keys for the question data, or to set the properties such as application type, version, permission for data files etc.

Note: Perl interpreter version 5.8.8 is pre-bundled with the Veritas Resiliency Platform.

The table below lists the key steps of working with APIs.

Table 2-1 Using Perl APIs

Description	Refer to
Create a Perl script "app.pl". This script must include the following modules: ■ Application.pm ■ Constants.pm	See "Using Perl APIs" on page 7.
Use the APIs to discovery and manage your applications.	The APIs are classified in the following categories. See "Define an application" on page 8. See "Logging" on page 12. See "Application inputs" on page 15. See "Application objects" on page 13. See "Application properties" on page 17.
Use this API to commit or finish your script.	See "Finish" on page 20.
See sample script and JSON output.	See "Sample script" on page 25. See "Sample script output" on page 31.
Test your scripts	See "Unit testing of your application module script" on page 23. See "Testing a script using CLI" on page 24.

Using Perl APIs

The APIs are classified in the following broad categories:

- Define an application
  This category consists of the *new* API which is the starting point of any application module script. It also contains the APIs that let you define the discovery level of the application, define the operations that can be performed on the application and so on.
- Logging
  Use the APIs in this category to log a message or reset the log levels.
- Application inputs
  Use the APIs in this category to add questions and their responses. These questions are displayed on the Resiliency Platform console.
- Application objects

Use the APIs in this category to add application instances and its sub-components. Application objects are application instances like server, main process, The application sub-components are middle tire servers, databases and its files like data files, configuration files, etc.

- Application properties
  Use the APIs in this category to define and retrieve application properties.
- Finish
  The only API in this category is *commit*, which indicates the end of application module script.
  The API generates a well-formed JSON output containing all the required information about the application.

Define an application

This category consists of the *new* API which is the starting point of any application module script. It also contains the APIs that let you define the discovery level of the application, define the operations that can be performed on the application and so on.

new

Description: This API conveys the application type, for example MSSQL, Oracle, SAP, to the Resiliency Platform. This interface is the starting point of any application module script and is mandatory. This initiates the application along with the logger. The logging level is set to **Info** by default.

To change the log level, use the *reset_log* API.

See "Logging" on page 12.

Note: This interface returns an un-defined object if user specified application type has anything other than the characters A-Z,a-z,0-9,dash(-), or underscore(_).

Is mandatory: Yes

Input parameter: String : Application type.

Return value: Application object, else undefined.

Syntax:

```
new VRTS::AppSDK::AppEnablementSDK::Application("<Application Type>");
```

Example:

```
my $appObj = new VRTS::AppSDK::AppEnablementSDK::Application("TestApp");
```

set_discovery_types

Description: Use this interface to declare the types of discoveries that your application script supports. For the Resiliency Platform to discover the applications, you need to define the discovery types such as deep and probe. Use comma as a delimiter to define more than one discovery level.

Deep and probe discovery types are mandatory.

- DEEP: discovers the entire application and its components including files.
- PROBE: only checks the status of the application instances. For example whether the application is online or offline.

Is mandatory: Yes

Input parameter: String : Discovery level type, value must be "DEEP" and "PROBE".

Return value: 0 if successful, else any positive number.

Example:

```
my $appObj = new VRTS::AppSDK::AppEnablementSDK::Application("TestApp");
$appObj->set_discovery_types("DEEP", "PROBE");
```

set_operations_types

Description: Use this interface to provide a list of operations that a particular application module script supports. For example, start an application and stop an application. Use comma as a delimiter to define more than one operation type. This interface is mandatory for the Resiliency Platform to execute the discovery script and perform operations on the applications.

Start and stop operations are mandatory.

Is mandatory: Yes

Input parameter: String: Operations types, value must be "START" and "STOP".

Return value: 0 if successful, else any positive number.

Example:

```
my $appObj = new VRTS::AppSDK::AppEnablementSDK::Application("TestApp");
$appObj->set_operation_types("START", "STOP");
```

register_discovery_callback

Description: Use this interface to register the discovery operation callback function against the discovery operation type defined using the *set_discovery_types* API. The registered callback function is invoked only when the application module script is invoked with said discovery operation. This is a mandatory interface that conveys

the Resiliency Platform the sub-routine which is capable of executing the operation and returning an appropriate return code and return message.

Ensure that the discovery script continues to discover the application in offline mode. Else, when the application is offline it is not discovered, and hence the instance is removed from the Resiliency Platform console. The resiliency group created using the instance becomes invalid.

The callback function is a sub-routine which is defined in the application module script and is capable of executing a said discovery operation successfully. The callback function returns 0 if the said operation is executed successfully else returns any positive number to indicate failure. Along with the return code you can also return a string containing either the success or the failure message. Providing a return code is mandatory otherwise the operation is considered as failed. Providing a return message is optional.

Is mandatory: Yes

Input parameter: String: Discovery operation - the discovery operation name which is previously set using the *set_discovery_types* API.

Input parameter: Callback function reference - the reference of the sub-routine which is defined in the application module script.

Return value: 0 if successful, else any positive number.

Example:

```
my $appObj = new VRTS::AppSDK::AppEnablementSDK::Application("TestApp");
if(defined $appObj)
{
 $appObj->set_discovery_types("DEEP, PROBE");
 $ret=$appObj->register_discovery_callback("PROBE",\&probe);
 $ret=$appObj->register_discovery_callback("DEEP",\&deep);
}

sub probe
{

 # Write the code here to discover all application instances

 my $inst_name = "app_inst";
 $appObj->log(LOGLEVEL_DEBUG,"Application instance name: [$inst_name]");

 # Write the code here to discover state of each application instance

 # Report the state of the discovered application instance
```

```
 # on the Resiliency Platform.
 my $inst = $appObj->add_application_inst($inst_name);
 if (defined $inst)
 {
  # The state must be reported either 'online' or 'offline'
  $inst->set_property(APP_INST_STATE, "Online");
 }

 # return 0 for successful and 1 for failure
 return 0,"probe is successful";
}
```

Note: The second parameter in the above example, "\&probe" and "\&deep", is the reference of the callback function i.e the sub-routine reference defined in the application module script.

---

register_operation_callback

Description: Use this interface to register the callback function against the operation type defined using the *set_operation_types* API. The registered callback function is invoked only when the application module script is invoked with said operation. This is a mandatory interface that conveys the Resiliency Platform the sub-routine which is capable of executing the operation and returning an appropriate return code and return message.

The callback function is a sub-routine which is defined in the application module script and is capable of executing a said operation successfully. The callback function returns 0 if the said operation is executed successfully else returns any positive number to indicate failure. Along with the return code you can also enter a string containing either the success or the failure message. Providing a return code is mandatory otherwise the operation is considered as failed. Providing a return message is optional.

The callback function when registered with AppEnablementSDK using any of the above APIs receives an hashref as a parameter. The hashref parameter contains `INSTANCE_NAME` as a key and application instance name as a value.

Is mandatory: Yes

Input parameter: String : Operation – The operation name which is previously defined using *set_operation_types* API.

Callback function reference - the reference of the sub-routine which is defined in the application module script.

Return value: 0 if successful, else any positive number.

Example:

```
my $appObj=new VRTS::AppSDK::AppEnablementSDK::Application("TestApp");
if(defined $appObj)
{
 $appObj->set_operation_types("START,STOP");
 $ret = $appObj->register_operation_callback("START",\&start);
 $ret = $appObj->register_operation_callback("STOP",\&stop);
}

sub start
{
 my ($arg) = @_;

 my $FuncName = ( caller 0 )[3];
 $appObj->log(LOGLEVEL_DEBUG,"Inside $FuncName");

 my $inst_name = $arg->{INSTANCE_NAME};
 $appObj->log(LOGLEVEL_DEBUG,"Application instance name: [$inst_name]");

 # Write the code here to start an application instance
 # and return appropriate status code and message.
 # Return 0 for successful and 1 for failure

 return 0, "Start is successful";
}
```

Note: The second parameter in the above example, "\&start" and "\&stop", is the reference of the callback function i.e the sub-routine reference defined in the application module script.

Logging

Use the APIs in this category to log a message or reset the log levels.

reset_log

Description: This interface is used to reset the logger to another user defined log level.

When you run the *new* interface, the log level is set to Info. Use the *reset_log* interface to change the log level to any one of the following:

- error

- debug
- warning
- critical
- trace

Is mandatory: No

Input parameter: String: log level

Return value: 0 if successful and the log level is reset to the new value, else any positive number.

Example:

```
$appObj->reset_log("debug");
```

log

Description: This interface is used to log messages into a log file.

If either of the log level or log message is empty, then the log message is not logged into the log file.

If you specify an invalid log level, that is anything other than error, debug, info, warning, critical, trace, then all such logs are logged with the log level as **error**.

The location of the log file is as below:

- Windows: `c:\ProgramData\Symantec\VRTSsfmh\APP\log\APP_TYPE.log`
- Linux: `/var/opt/VRTSsfmh/APP/log/APP_TYPE.log`

Is mandatory: No

Input parameter: String: Log level such as error, debug, warning, critical, or trace, and the log message

Return value: NA

Example:

```
$appObj->log("error","test message....");
```

Application objects

Use the APIs in this category to add application instances, its sub-components such as the middle tier servers, databases, and the support files such as the configuration file, database file.

add_application_instance

An application instance comprises of the application server or the application main process. Reporting the application instances is mandatory so as to view the applications in the **Unmanaged** tab on the Resiliency Platform web console and to perform operations on them.

Use this interface to create an application instance object which is used to add application instance specific properties like application instance name, version, home directory. To do this use the *set_property* API. Defining the application state property, that is **State**, is mandatory to perform any further operations on the application using the Resiliency Platform console.

Is mandatory: Yes

Input parameter: String: The name of the application instance.

Return value: 0 and the application instance object, else any positive number.

Example:

```
my $appInstObj = $appObj->add_application_inst ("test_inst");
```

add_application_unit

An application unit comprises of the application sub-components such as the middle tier servers, databases, etc which support the application instance function. Reporting these components is optional. They are not displayed on the Resiliency Platform console and hence you cannot perform any operations on them using the console. But if you want to perform the start application or stop application operation at the component level, you need to define these components.

Use this interface to create an application unit object which is used to add application unit specific data like application database name.

Is mandatory: No

Input parameter: String: The name of the application unit.

Return value: 0 and the application unit object, else any positive number.

Example:

```
my $appUnit1Obj = $appInstObj->add_application_unit("master_db");

my $appUnit2Obj = $appInstObj->add_application_unit("temp_db");
```

add_application_file

An application file comprises of application support files such as the configuration file, database file etc. Reporting the application files is mandatory to be able to perform disaster recovery operations using the Resiliency Platform console.

Use this interface to create an application file object which is used to add application file specific data like application file name, size, path.

Defining the application property, **Type**, is mandatory if you want to configure the application for disaster recovery using the Resiliency Platform console

Is mandatory: Yes

Input parameter: String: The name of the application file or file path.

Return value: 0 and the application file object, else any positive number.

Example:

```
my $appFileObj = $appInstObj->add_application_file("master.txt");

my $appFileObj = $appInstObj->add_application_file("temp.txt");
```

Application inputs

Use the APIs in this category to add questions and their responses. These questions are displayed on the Resiliency Platform console.

add_question

Description: Use this interface to add questions to complete the discovery of an application instance.

These questions are displayed on the Resiliency Platform console if the application instance is partially discovered and user inputs are required to complete the discovery.

The following table lists the questions that you can add.

Table 2-2 Questions and error messages

Question data field	Description	Expected value
QID	Message ID for the question text. A whole number denoting the question number.	Any positive number. This is a mandatory field.
QText	The question text.	Any text. This is a mandatory field.
QDescription	Description of the question.	Any text. This is an optional field.
Mandatory	Define this if an answer is mandatory to the question.	Yes or no. This is a mandatory field.

Table 2-2 Questions and error messages *(continued)*

Question data field	Description	Expected value
IsError	Define this if an error occurs for a question.	Yes or no. This is an optional field.
Encrypted	Define this if the answer needs encryption.	Yes or no. This is an optional field. Is set to “no” by default.
ErrorCode	Error code of the error message. If the response received is incorrect user needs to set this field.	Any positive number. This is an optional field.
ErrorMsg	Error message. If the response received is incorrect user needs to set this field.	Any text. This is an optional field. Between an error code and an error message any one must be mentioned in case of error. An error message is preferred.

Is mandatory: No

Input parameter: Application instance object, question data

Return value: NA

Example:

```
my $qid1 = {
  'QID' => 1,
  'QDescription' => 'Specify the administrator user name to
    start the above instance.',
    'Mandatory' => 'yes',
  'QText' => 'Administrator user name for this instance',
  'Encrypted' => 'no'
};
$appInstObj->add_question($qid1);
```

get_qresponse

Description: This interface returns the response to the questions which you have defined using the *add_question* API. You can call this API when you need responses to the questions. The responses assist in completing the application discovery.

Is mandatory: No

Input parameter: String: Question ID

Return value: Response to the question ID if successful, else undefined.

Example:

```
$appInstObj->get_qresponse($qid);
```

Where $qid is the question ID.

set_qresponse

Description: Use this interface to define an error code and error message in response to any error that occurs for a particular question ID.

Is mandatory: No

Input parameter: String: Error code, error message, question ID

Return value: 0 with question data set with an error code and error message, else any positive number.

Example:

```
$appInstObj->set_qresponse($error_code, $error_string, $qid);
```

Application properties

Use the APIs in this category to define and retrieve application properties.

set_property

Description: Use this interface to set the predefined properties for an application type object. Application type objects are application instance, application unit, and application file.

If you want to set custom or user defined properties, use the *set_custom_property* API.

The below table lists the properties for application instance, unit, and file.

Table 2-3

Property name	Description	Expected value	Comments
VERSION	Version number	Any valid string value.	
OWNER	Owner name	Any valid string value.	
STATE	State	Online or offline.	This property is mandatory for application instance.

Table 2-3 *(continued)*

Property name	Description	Expected value	Comments
IS_PARALLEL	Whether the application supports parallel instances.	Yes or no.	If your application supports parallel instances then set this property to **yes** otherwise **no**. This is applicable for application instance and unit.
HOMEDIR	Home directory	Any valid string value.	This is applicable for application instance and unit.
TOTAL_SIZE	Total size	Any valid string value.	This is applicable for application instance and unit.
USED_SIZE	Used size	Any valid string value.	This is applicable for application instance and unit.
TYPE	Type	Any valid string value.	This is applicable for application unit and file. For application file, set this property to **Data** if you want to the Resiliency Platform to consider this file for disaster recovery (DR) configuration.
SIZE	Size	Any valid string value.	This is applicable only for application file.
PERMISSION	Permission of an application file.		This is applicable only for application file.

Table 2-3 *(continued)*

Property name	Description	Expected value	Comments
FILE_PATH			This is applicable only for application file. Set this property to full path of an application file if you want the Resiliency Platform to consider this file for DR configuration.

Is mandatory: No

Input parameter: String: Attribute name and attribute value

Return value: 0 if successful, else any positive number.

Example:

```
Application instance:
my $inst = $appObj->add_application_inst("app_inst");
if (defined $inst)
{
  $inst->set_property("FRIENDLY_NAME", "MyAppInstance");
}

Application unit:
my $inst = $appObj->add_application_inst("app_inst");
if (defined $inst)
{
  my $unit = $inst->add_application_unit("app_unit");
  $unit->set_property("OWNER", "MyOwner");
}

Application file:
my $inst = $appObj->add_application_inst("app_inst");
if (defined $inst)
{
my $file = $inst->add_application_file("master.data");
$file->set_property("SIZE", "100");
}
```

set_custom_property

Description: Use this interface to define custom property of your choice.

You can use this interface to set custom defined properties for an application instance, application unit, and application file.

Is mandatory: No

Input parameter: String: Attribute name, attribute value, is_secure (true or false), and attribute type. Attribute is_secure and type are optional. If is_secure is set to true, then property value is encrypted.

Return value: 0 if property is added successfully, else any positive number.

Example:

```
Application instance:
my $inst = $appObj->add_application_inst("app_inst");
if (defined $inst)
{
  $inst->set_custom_property("HOMEDIR", "MyDir");
}

Application unit:
my $inst = $appObj->add_application_inst("app_inst");
if (defined $inst)
{
  my $unit = $inst->add_application_unit("app_unit");
  $unit->set_custom_property("USER", "MyUser");
}

Application file:
my $inst = $appObj->add_application_inst("app_inst");
if (defined $inst)
{
my $file = $inst->add_application_file("master.data");
$file->set_custom_property("CHECKSUM", "123456789");
}
```

Finish

The only API in this category is *commit*, which indicates the end of application module script.

commit

Description: Use this interface to indicate the end of application module script. A JSON output is generated which is used by the Application Enablement SDK. The output consists of a number of tags and data which was provided while running the script. Call this interface with **print** to print a JSON output on a STDOUT.

Is mandatory: Yes

Input parameter: None

Return value: A JSON output

Example:

```
print ($appObj->commit());
```

Managing clustered applications

The Resiliency Platform lets you manage the applications that are clustered using any high availability technology. You can manage the applications as well as view the clustering technology details on the Resiliency Platform web console. To do this you must set the following custom properties using the *set_custom_property* API on the application instance object.

Table 2-4 Custom property names

Custom property name	Description	Expected value
ClusterType	Type of your clustering technology. This value is displayed on the web console. e.g. MSCS.	Any valid string value. e.g. MSCS.
ServiceGroupName	Name of the container or the service group name in the cluster.	Any valid string value.
IsClustered	Indicates whether your application is clustered or not.	Valid string value. **true** if application is clustered, else **false**.

Ensure that you set all the three properties. Else, Resiliency Platform treats the application as non-clustered and separate entries for the same application are displayed on the console.

To view the clustering technology name on the console

1 Navigate

Assets > **Unmanaged** tab.

2 Select **Application** in **Asset Type**. The name is displayed in the **Availability** column.

Sample script to set to custom properties

```
my $inst = $appObj->add_application_inst("sample_inst");
if (defined $inst)
{
  $inst->set_custom_property("ClusterType", "MSCS");
  $inst->set_custom_property("ServiceGroupName", "sample_inst_group");
  $inst->set_custom_property("IsClustered", "true");
}
```

Chapter 3

Testing the APIs

This chapter includes the following topics:

Unit testing of your application module script

The application module script is invoked as follows:

```
<Script_name>.pl -aes_args <args_file_path>
```

`args_file` contains the following:

For probe and deep discovery -

```
"{\"ARGS\":{\"AES_ARGS\":{\"OP_TYPE\":\"DISCOVERY\",\"OP\":\"PROBE\"}}}"
```

For Start and Stop operation -

```
"{\"ARGS\":{\"AES_ARGS\":{\"OP_TYPE\":\"OPERATION\",\"OP\":\"START\",
\"APP_INST_ID\":\"sdkdb\"}}}"
```

- OP_TYPE : This is the operation type. Value is either 'DISCOVERY' or 'OPERATION'.
- OP: This is the operation that is to be performed. Value can be 'PROBE', 'DEEP', 'START' and 'STOP'.
- AAP_INST_ID : This is the application instance name on which the operation is to be performed. This is optional for 'PROBE' and 'DEEP' discovery but mandatory for 'START' and 'STOP' operation.

Testing a script using CLI

Use the following steps to test the application module script on the managed host using the CLI.

Testing a script using CLI

1. Copy the application module script to any location on the managed host.
2. Open command prompt on Windows or shell in case of Linux.
3. Create an argument file and copy the following content in it .

   For probe and deep discovery:

   ```
   {"ARGS":{"AES_ARGS":{"OP_TYPE":"DISCOVERY","OP":"PROBE"}}}"
   ```

   For start and stop operation:

   ```
   {"ARGS":{"AES_ARGS":{"OP_TYPE":"OPERATION","OP":"START","APP_INST_ID"
   :"sdkdb"}}}
   ```

4. Change the value of the JSON tags “OP_TYPE" and "OP” as per the implementation of the application module script.

   OP_TYPE supported values are “OPERATION” or “DISCOVERY".

   OP value could be the operation that you have registered and implemented, such as probe, start, etc.

5. Invoke the script with argument —aes_args and the argument file that you have created.

   For example: sample_app.pl —aes_args <args_file_path>

6. Check the log and the command output on STDOUT.

   Command output varies as per the operation you have mentioned in “OP” in the args file.

   The location of the log file is as below:

   - Windows: `c:\ProgramData\Symantec\VRTSsfmh\APP\log\APP_TYPE.log`
   - Linux: `/var/opt/VRTSsfmh/APP/log/APP_TYPE.log`

Sample script

Find below a template script that you can use for developing your application module script. A sample script is also provided with hard coded data that can be modified to develop your script.

Sample script

```
use strict;
use warnings;
use VRTS::AppSDK::AppEnablementSDK::Application;
use VRTS::AppSDK::AppEnablementSDK::Constants qw(:LOG_LEVELS :
APP_INST_ATTRS :APP_UNIT_ATTRS :APP_FILE_ATTRS :
APP_CUSTOM_ATTRS :APP_INPUT_KEYS);

# Define an application here
my $appObj = new VRTS::AppSDK::AppEnablementSDK::Application("SampleApp");
if(defined $appObj)
{
  # Set and register various operations that this application script
  # supports.
  $appObj->set_discovery_types("DEEP, PROBE");
  $appObj->set_operation_types("START, STOP");

  $appObj->register_operation_callback("START",\&start);
  $appObj->register_operation_callback("STOP",\&stop);
  $appObj->register_discovery_callback("PROBE",\&probe);
  $appObj->register_discovery_callback("DEEP",\&deep);

  print ($appObj->commit());
}
################################
#Function: start
#
#Starts an application instance
#
#Parameters:
#arg - hash containing the application instance name
#
#Returns:
# 0 if successful else 1
# you can also return success or failure message which is optional.
################################
```

```
sub start
{
  my ($arg) = @_;

  my $FuncName = ( caller 0 )[3];
  $appObj->log(LOGLEVEL_DEBUG,"Inside $FuncName");

  my $inst_name = $arg->{INSTANCE_NAME};
  $appObj->log(LOGLEVEL_DEBUG,"Application instance name: [$inst_name]");

  #Write the code here to start an application instance
  #and return an appropriate status code and message.
  #Return 0 for success and 1 for failure.

  return 0, "Start is successful";
}

################################
#Function: stop
#
#Stops an application instance
#
#Parameters:
#arg - hash containing the application instance name
#
#Returns:
# 0 if successful else 1
# you can also return success or failure message which is optional.
################################

sub stop
{
  my ($arg) = @_;

  my $FuncName = ( caller 0 )[3];
  $appObj->log(LOGLEVEL_DEBUG,"Inside $FuncName");

  my $inst_name = $arg->{INSTANCE_NAME};
  $appObj->log(LOGLEVEL_DEBUG,"Application instance name: [$inst_name]");

  #Write the code here to stop an application instance
  #and return an appropriate status code and message.
  #Return 0 for success and 1 for failure.
```

```
  return 0, "Stop is successful";
}
#################################
#Function: probe
#
#Discovers and reports the application instance state
#
#Parameters:
#arg - hash containing the application instance name
#
#Returns:
# 0 if successful else 1
# you can also return success or failure message which is optional.
#################################

sub probe
{
  my ($arg) = @_;

  my $FuncName = ( caller 0 )[3];
  $appObj->log(LOGLEVEL_DEBUG,"Inside $FuncName");

  #Write the code here to discover and report the application instance name.

  #Report the state of the discovered application instance
  # on the Resiliency Platform.

  my $inst_name = $appObj->add_application_inst("app_inst");
  if (defined $inst)
  {
    # The state must be reported either 'online' or 'offline'
    $inst_name->set_property(APP_INST_STATE,"Online");
  }

  #return 0 if successful else 1
  return 0, "Probe is successful";
}
#################################
#Function: deep
#
#Discovers and reports the sub-components and data file information of
#an application instance.
```

```
#
#Parameters:
#arg - hash containing the application instance name
#
#Returns:
# 0 if successful else 1
# you can also return success or failure message which is optional.
#################################

sub deep
{
  my ($arg) = @_;

  my $FuncName = ( caller 0 )[3];
  $appObj->log(LOGLEVEL_DEBUG,"Inside $FuncName");

  #Write the code here to discover and report the sub-components
  # and the data file information of an application instance.

  my $inst_name = $appObj->add_application_inst("app_inst");
  if (defined $inst)
  {
    # Set application instance properties
    # Ensure that the following property is set, else the application
    # state is not displayed on the Resiliency Platform web console.
    # You cannot perform operations if the state is not displayed
    # on the console.
    # Accepted values are 'online' and 'offline'

    $inst_name->set_property(APP_INST_STATE,"Online");

    # Following properties are optional

    $inst->set_property(APP_INST_VERSION,"1.0");
    $inst->set_property(APP_INST_OWNER,"Administrator");
    $inst->set_property(APP_INST_ISPARALLEL,"false");
    $inst->set_property(APP_INST_HOMEDIR,"inst_homedir");
    $inst->set_property(APP_INST_APP_TYPE,"emp_database");
    $inst->set_property(APP_INST_APP_CATEGORY,"database");
    $inst->set_property(APP_INST_TOTAL_SIZE,"100");
    $inst->set_property(APP_INST_USED_SIZE,"90");

    # Set application instance custom properties
```

```
# Setting custom properties is optional.

$inst->set_custom_property("app_disp_name", "sample_instance");

################################
# If your application is clustered using any high availability
# technology, then you need to set the following custom properties.

# $inst->set_custom_property("ClusterType", "MSCS");
# $inst->set_custom_property("ServiceGroupName", "sample_sg");
# $inst->set_custom_property("IsClustered", "true");

#-------------------------------

# Check if your application module script requires additional
# information such as user name and password from the user
# who is accessing the Resiliency Platform web console.

# If information is required, then check if the information
# is already asked and do we have its responses available here
# using the following API. Use those responses to complete
# your task.

# Use following API with QID as input
  my $response1 = $inst->get_qresponse('1');
  my $response2 = $inst->get_qresponse('2');
#-------------------------------

# If response is not available then ask for information
# again using the following API:

# Define questions to be asked in a hash
  my $qid1 = {
     'QID' => '1',
     'QText' => 'Administrator user name',
     'Mandatory' => 'yes',
     'QDescription' => 'Specify the administrator user name
     to discover its data files.',
     'Encrypted' => 'no'};

  my $qid2 = {
     'QID' => '2',
```

```
     'QText' => 'Administrator password',
     'Mandatory' => 'yes',
     'QDescription' => 'Specify the administrator user password
     to discover its data files.',
     'Encrypted' => 'yes'};

 # Add the hash using the following API:
 $inst->add_question($qid1);
 $inst->add_question($qid2);
 #-------------------------------

# Write the code here to discover an application unit (application
# sub-components) and application files information.
# Note: Discovering application unit is not mandatory
# but discovering application files is mandatory.

# Add application sub-component here
my $unit = $inst->add_application_unit("app_unit");

# Add properties for application unit
# Following properties are optional:

$unit->set_property(APP_UNIT_OWNER, "unit_owner");
$unit->set_property(APP_UNIT_VERSION, "1.0");
$unit->set_property(APP_UNIT_STATE, "online");
$unit->set_property(APP_UNIT_ISPARALLEL, "false");
$unit->set_property(APP_UNIT_HOMEDIR, "unit_homedir");
$unit->set_property(APP_UNIT_TYPE, "database");
$unit->set_property(APP_UNIT_TOTAL_SIZE, "100");
$unit->set_property(APP_UNIT_USED_SIZE, "50");

# Setting custom properties is optional.
$unit->set_custom_property("unit_disp_name", "sample_unit");
#-------------------------------

# Write the code here to discover information of application data files.

my $file = $inst->add_application_file("app_file");

# Following property is important and mandatory if you want to
# configure your application for disaster recovery.
# Value of this property could be data, log, etc. but the
# Resiliency Platform considers only those DR configuration files
```

```
    # which are marked as 'data'.

    $file->set_property(APP_FILE_TYPE, "data");

    # Following property is important and mandatory if you want to
    # configure your application for disaster recovery.

    # Value of the property must be full file path
    # e.g. '/root/app_inst/app_file.data' or
    # 'c:\\app_inst\\app_file.data'

    $file->set_property(APP_FILE_PATH, "c:\\app_inst\\app_file.data");

    # Following properties are optional:

    $file->set_property(APP_FILE_NAME, "app_file.data");
    $file->set_property(APP_FILE_VERSION, "1.0");
    $file->set_property(APP_FILE_OWNER, "administrator");
    $file->set_property(APP_FILE_STATE, "online");
    $file->set_property(APP_FILE_SIZE, "10");
    $file->set_property(APP_FILE_PERMISSION, "all");

    # Setting custom properties is optional.

    $file->set_custom_property("file_desc", "Database file");

  }

   #return 0 if successful else 1
   return 0, "deep discovery is successful";
}
```

Sample script output

Below are outputs of some sample scripts with different use cases.

Scenario 1: Probe Discovery

Args input file contains:

```
{"ARGS":{"AES_ARGS":{"OP_TYPE":"DISCOVERY","OP":"Probe"}}}
```

JSON output after script execution:

```
{
   "APPLICATION" : {
      "APP_NAME" : "SampleApp",
      "OPERATION_TYPES" : {
         "START" : {
            "Name" : "START"
         },
         "STOP" : {
            "Name" : "STOP"
         }
      },
      "DISCOVERY_TYPES" : {
         "DEEP" : {
            "Name" : "DEEP"
         },
         "PROBE" : {
            "Name" : "PROBE"
         }
      },
      "AES_VERSION" : "1.0.0.0"
   },
   "APPLICATION_INSTANCE" : {
      "SampleApp" : {
         "APP_ID" : "SampleApp",
         "NAME" : "SampleApp",
         "DISCOVERY_TYPE" : "discovered",
         "INFO_REQUIRED" : "no",
         "STATE" : "Online",
         "APP_TYPE" : "SampleApp"
      }
   },
   "OPERATIONS" : {
      "PROBE" : {
         "OPERATION_NAME" : "PROBE",
         "APP_NAME" : "SampleApp",
         "OPERATION_ARGS" : {
            "INSTANCE_NAME" : "SampleApp"
         },
         "OPERATION_EXECUTED" : 1,
         "RET_CODE" : 0,
         "AES_VERSION" : "1.0.0.0",
         "OPERATION_TYPE" : "DISCOVERY",
         "ERR_CODE" : 0,
```

```
         "RET_MSG" : "probe is successful"
      }
   }
}
```

Scenario 2: Deep Discovery

Args input file contains:

```
{"ARGS":{"AES_ARGS":{"OP_TYPE":"DISCOVERY","OP”:"DEEP"}}}
```

JSON output after script execution:

```
{
   "APPLICATION" : {
      "APP_NAME" : "SampleApp",
      "OPERATION_TYPES" : {
         "START" : {
            "Name" : "START"
         },
         "STOP" : {
            "Name" : "STOP"
         }
      },
      "DISCOVERY_TYPES" : {
         "DEEP" : {
            "Name" : "DEEP"
         },
         "PROBE" : {
            "Name" : "PROBE"
         }
      },
      "AES_VERSION" : "1.0.0.0"
   },
   "APPLICATION_INSTANCE" : {
      "app_inst" : {
         "APP_ID" : "app_inst",
         "NAME" : "app_inst",
         "HOMEDIR" : "inst_homedir",
         "APPLICATION_INSTANCEProps" : {
            "SERVICEGROUPNAME" : {
               "PROP_TYPE" : "",
               "PROP_VALUE" : "sample_sg",
               "PROP_NAME" : "ServiceGroupName"
            },
```

```
      "CLUSTERTYPE" : {
         "PROP_TYPE" : "",
         "PROP_VALUE" : "MSCS",
         "PROP_NAME" : "ClusterType"
      },
      "ISCLUSTERED" : {
         "PROP_TYPE" : "",
         "PROP_VALUE" : "true",
         "PROP_NAME" : "IsClustered"
      },
      "APP_DISP_NAME" : {
         "PROP_TYPE" : "",
         "PROP_VALUE" : "sample_instance",
         "PROP_NAME" : "app_disp_name"
      }
   },
   "TOTAL_SIZE" : 100,
   "DISCOVERY_TYPE" : "discovered",
   "APP_INPUTS" : {
      "QID1" : {
         "QID" : 1,
         "QDescription" : "Specify the administrator user name
          to discover its data files.",
         "Mandatory" : "yes",
         "QText" : "Administrator user name",
         "Encrypted" : "no"
      },
      "QID2" : {
         "QID" : 2,
         "QDescription" : "Specify the administrator user password
          to discover its data files.",
         "Mandatory" : "yes",
         "QText" : "Administrator password",
         "Encrypted" : "yes"
      }
   },
   "APP_CATEGORY" : "database",
   "INFO_REQUIRED" : "yes",
   "APP_TYPE" : "SampleApp",
   "STATE" : "online",
   "VERSION" : 1,
   "USED_SIZE" : 90,
   "OWNER" : "Administrator"
```

```
   }
},
"APPLICATION_FILE" : {
   "app_inst;app_file" : {
      "SIZE" : 10,
      "APP_ID" : "app_inst",
      "NAME" : "app_file",
      "APPLICATION_FILEProps" : {
         "FILE_DESC" : {
            "PROP_TYPE" : "",
            "PROP_VALUE" : "Database file",
            "PROP_NAME" : "file_desc"
         }
      },
      "TYPE" : "data",
      "PERMISSION" : "all",
      "FILE_PATH" : "c:\\app_inst\\app_file.data",
      "FILE_ID" : "app_inst;app_file",
      "STATE" : "online",
      "VERSION" : 1,
      "OWNER" : "administrator"
   }
},
"APPLICATION_UNIT" : {
   "app_inst;app_unit" : {
      "APP_ID" : "app_inst",
      "APPLICATION_UNITProps" : {
         "UNIT_DISP_NAME" : {
            "PROP_TYPE" : "",
            "PROP_VALUE" : "sample_unit",
            "PROP_NAME" : "unit_disp_name"
         }
      },
      "NAME" : "app_unit",
      "HOMEDIR" : "unit_homedir",
      "TYPE" : "database",
      "APP_UNIT_ID" : "app_inst;app_unit",
      "TOTAL_SIZE" : 100,
      "STATE" : "online",
      "VERSION" : 1,
      "USED_SIZE" : 50,
      "OWNER" : "unit_owner"
   }
```

```
   },
   "OPERATIONS" : {
      "DEEP" : {
         "OPERATION_NAME" : "DEEP",
         "APP_NAME" : "SampleApp",
         "OPERATION_EXECUTED" : 1,
         "RET_CODE" : 0,
         "AES_VERSION" : "1.0.0.0",
         "OPERATION_TYPE" : "DISCOVERY",
         "ERR_CODE" : 0,
         "RET_MSG" : "deep discovery is successful"
      }
   }
}
```

Scenario 3: Start operation

Args input file contains:

```
{"ARGS":{"AES_ARGS":{"OP_TYPE”:"OPERATION","OP”:"START","APP_INST_ID"
:"SampleApp"}}}
```

JSON output after script execution:

```
{
   "APPLICATION" : {
      "APP_NAME" : "SampleApp",
      "OPERATION_TYPES" : {
         "START" : {
            "Name" : "START"
         },
         "STOP" : {
            "Name" : "STOP"
         }
      },
      "DISCOVERY_TYPES" : {
         "DEEP" : {
            "Name" : "DEEP"
         },
         "PROBE" : {
            "Name" : "PROBE"
         }
      },
      "AES_VERSION" : "1.0.0.0"
   },
```

```
   "OPERATIONS" : {
      "START" : {
         "OPERATION_NAME" : "START",
         "APP_NAME" : "SampleApp",
         "OPERATION_ARGS" : {
            "INSTANCE_NAME" : "SampleApp"
         },
         "OPERATION_EXECUTED" : 1,
         "RET_CODE" : 0,
         "AES_VERSION" : "1.0.0.0",
         "OPERATION_TYPE" : "OPERATION",
         "ERR_CODE" : 0,
         "RET_MSG" : "Start is successful"
      }
   }
}
```


Scenario 4: Stop operation

Args input file contains:


```
{"ARGS":{"AES_ARGS":{"OP_TYPE”:"OPERATION","OP”:"STOP","APP_INST_ID"
:"SampleApp"}}}
```


JSON output after script execution:


```
{
   "APPLICATION" : {
      "APP_NAME" : "SampleApp",
      "OPERATION_TYPES" : {
         "START" : {
            "Name" : "START"
         },
         "STOP" : {
            "Name" : "STOP"
         }
      },
      "DISCOVERY_TYPES" : {
         "DEEP" : {
            "Name" : "DEEP"
         },
         "PROBE" : {
            "Name" : "PROBE"
         }
      },
```

```
      "AES_VERSION" : "1.0.0.0"
   },
   "OPERATIONS" : {
      "STOP" : {
         "OPERATION_NAME" : "STOP",
         "APP_NAME" : "SampleApp",
         "OPERATION_ARGS" : {
            "INSTANCE_NAME" : "SampleApp"
         },
         "OPERATION_EXECUTED" : 1,
         "RET_CODE" : 0,
         "AES_VERSION" : "1.0.0.0",
         "OPERATION_TYPE" : "OPERATION",
         "ERR_CODE" : 0,
         "RET_MSG" : "Stop is successful"
      }
   }
}
```

Check the following properties in the OPERATIONS tag in the JSON output after executing a script.

- OPERATION_EXECUTED denotes whether the operation is executed. 1 indicates executed, 0 for not executed.
- RET_CODE denotes whether the operation is successfully executed. 0 indicates success and any positive number for failure. This return code is set by the callback function for the respective operations.
- ERR_CODE displays the error code when the operation fails. 0 indicates no error.
- RET_MSG displays the return message for the respective operation. This return message is set by the callback function for the respective operations.

Chapter 4

Deployment

This chapter includes the following topics:

About the manifest file

The Application Enablement SDK bundle file should contain a manifest file named `bundle.ini`. A single bundle file can be used to create an add-on supporting multiple operating systems. The bundle file should have the following structure:

```
[Main]
name = my_app;
friendly name = My App;
category = Database;
vendor = My Company Inc;
description = My description;
version = 1.0.0.0;
copyright = Copyright (C) My Company Inc. All rights reserved.;

discovery_types = DEEP, PROBE;
operation_types = START, STOP;

[linux]
content = my_directory1;

[windows]
content = my_directory2;
```

The following table lists the descriptions of the fields.

Table 4-1 Descriptions

Field	Description
Name	Specify the name of the application.
Friendly name	Specify a friendly name which is displayed on the Resiliency Manager.
Category	Specify the application category such as database.
Vendor	Specify the name of the vendor.
Description	Specify a description of the application.
Version	Specify one to four dot-separated integers identifying the version. Integers must be between 0 and 999. Examples of version are: 1.2.3.4.
Copyright	Specify the copyright year.
osname	Specify Linux or Windows. Create separate sections for each of the supported operating systems. Linux - RHEL 6 x86_64 and RHEL 7 x86_64. Windows - All supported Windows x64 platforms.
content	This directory must contain the `app.pl` script. All the contents of this directory shall be a part of the add-on. The directory path specified in the content should be relative to the location of the `bundle.ini` file.

The directory structure to create the bundle must be as follows:

```
my_bundle_data
mybundle.tar.gz
|--bundle.ini
|--my_directory1
|   |--app.pl
|--my_directory2
    |--app.pl
```

You can create the bundle using the following command:

```
# tar czvf mybundle.tar.gz -C my_bundle_data/
```

You can verify that the bundle has been created as per the expected directory structure by executing the following command:

```
# tar tvf mybundle.tar.gz
```

Output should be as below:

```
mybundle.tar.gz
|--bundle.ini
|--my_directory1
|   |--app.pl
|--my_directory2
    |--app.pl
```

Note: The only supported format of the bundle is `.tar.gz`. For Windows, use any third party application to create the bundle in the `.tar.gz` format.

Index