IBM Application Runtime Expert for i

Script Interfaces to ARE

ARE Core version: 1.6.6

Document version: 1.1

For the latest information, visit ARE home site:
http://www.ibm.com/systems/power/software/i/are/index.html

Background

The IBM Application Runtime Expert for i core, or ARE core, is the ARE component that verifies a system comparing the values found on this system to the specified template. After creating a template using the deployment template editor, there are two ways to use that template to verify a system:

1. Use the Web interface from the ARE console
2. Use a script that can be run from QShell on the target system

The second option, using a script that can be run from QShell, is possible because of the script interfaces that are part of the ARE core and is shipped as part of the base operating system and is available via PTF on IBM i 5.4 and newer releases. These scripts can be used for tasks such as verifying a system or testing a custom plugin. This document describes the details of each script, including required and optional parameters, default values, and examples of using each script.

Before getting into the details of each script, there are a few things to be aware of when running ARE scripts. First, when you run an ARE script, it is run under the user profile of the signed on user. This means that if, for example, you use an ARE script to verify a system, the verification will all be done under your user profile. If the verification requires access to resources (files, user profiles, system values, etc) that you do not have authority to, the verification of those resources will fail. Therefore, it is typically recommended that verification be performed by a user that has *ALLOBJ special authority. There is, however, no hard requirement that the

user profile have *ALLOBJ special authority. This means that ARE will try to verify the system regardless of what special authorities the user has.

Typically the ARE scripts are simply a front end to the ARE core, which means the scripts do not directly output any messages or results. All output from the scripts is sent to the designated output files not the console. One notable exception to this is help text. All ARE scripts have help text available via the '-h' command line option.

IMPORTANT: All ARE scripts are located in the following directory:

/QIBM/ProdData/OS/OSGi/healthcheck/bin/

When running an ARE script in QShell, you should always use the fully qualified IFS path to the script.

runARE.sh

This script is the interface to start the ARE core and verify a system using a template that is provided as input.

Location

This script is located in /QIBM/ProdData/OS/OSGi/healthcheck/bin/

Syntax

The command syntax is as follows:

```
runARE.sh  -template template_name  -outFile|zipFile outfile_name  [-elementPath element_path] [-outXML xml_outfile_name] [-outSummary summary_outfile_name] [-property prop_key_value] [-version]
```

Parameters

- **-template *template_name*** – The name of the template that the ARE core should use to perform the verification. The template name must be the name of a Java jar file, and the name must be qualified such that the jar file can be found by the ARE core. **This is a required parameter**. If the specified deployment template cannot be found, a com.ibm.are.core.AreLaunchException exception is thrown by the ARE core.
- **-outFile *outfile_name*** – The name of the output file where the ARE core writes its detailed report. The ARE core uses this output file name as a base pattern for naming additional output files for other reports. For example, if you specify an output file name of “report.out”, the ARE core will produce the following output files:
  - report.out
  - report.out.summary.txt
  - report.out.xml

  Note: Either the -outFile or -zipFile parameter must be specified. If only a file name is provided for this parameter, all output files will be placed in the current working directory. If a fully qualified path and file name is provided, all output files will be placed in the location specified by the fully qualified path.
- **-zipFile *out_zip_file_name*** – The name of the output zip file where the ARE core writes its final packaged archive file. This parameter enables the Collector mode. The Collector mode allows all report files and other collectable files to be packaged into a single Zip file. This mode is especially useful when you are using the Resource Collector Plugin or Command Runner Plugin, which generates additional files.

- **-outXML *xml_outfile_name*** – The name of the output file where the ARE core writes its XML report. If this parameter is specified, the ARE core will use this file name for the XML report instead of generating a file name based on the output file name specified for the -outFile parameter. This is an optional parameter, and works only when –outFile is specified.
- **-outSummary *summary_outfile_name*** – The name of the output file where the ARE core writes its summary report. If this parameter is specified, the ARE core will use this file name for the summary report instead of generating a file name based on the output file name specified for the -outFile parameter. This is an optional parameter, and works only when –outFile is specified.
- **-property *prop_key_and_value*** – Specifies a Java property that should be passed to the JVM when the ARE core is started. This parameter can be repeated as many times as needed, allowing multiple properties to be passed to the JVM when the ARE core is started. Note that the *prop_key_and_value* must be the Java property key and value pair, separated by an equal sign. This is an optional parameter.

  Examples:

  -property my.prop.key=propValue

  -property are.log.level=FINE
- **-elementPath *element_path*** – Any additional jars that need to be added to the element path can be added using this property. The path should be in the same format as a Java classpath, with a single colon used to separate path entries. This is an optional parameter.

  Example:

  -elementPath /tmp/lib1.jar:/tmp/lib2.jar:/tmp/lib3.jar
- **-version** Display version of the ARE core on the current system. See the Usage Examples section for details.

Exit Codes

This script has a set of possible exit values that make it very easy to determine if the verification found any problems. This is most useful when invoking the runARE.sh script from another script; in this scenario, the exit value for the runARE.sh can be checked to determine if any problems were found, thus freeing the script author from having to try and parse and analyze the ARE reports to make that determination.

Possible exit values are:

0 – No problems at any severity level (error, warning, info) were found

5 – An exception occurred trying to launch the ARE core

6 – An exception occurred during the startup of the ARE core

7 – The ARE core encountered an unsupported or unrecognized input argument

8 – An auto start service failed to start

9 – The software products or versions necessary to verify the system using the provided template were not found

10 – The software products or versions necessary for the ARE core to run were not found on the system

49 – An exception occurred during the shutdown of the ARE core

50 – One or more problems were found. The highest severity problem reported was INFO.

60 – One or more problems were found. The highest severity problem reported was WARNING.

70 – One or more problems were found. The highest severity problem reported was ERROR.

255 – An unknown exception occurred within the ARE core

Usage Examples

For all of the examples below, we will use '/tmp/MyTemplate1.jar' as the template name.

1. Verify the system using the script with only the required parameters.

   QIBM/ProdData/OS/OSGi/healthcheck/bin/runARE.sh -template /tmp/MyTemplate1.jar -outFile verify.out

   The above example uses the MyTemplate1.jar template, located in the /tmp directory, to verify the system. The (detailed report) output will be written to the verify.out file. Additionally, a report summarizing only the detected problems will be written to the verify.out.summary.txt file. Finally, an XML version of the detailed report will be written to the verify.out.xml file.

2. Verify the system, overriding the name of the XML and summary report files.

   QIBM/ProdData/OS/OSGi/healthcheck/bin/runARE.sh -template /tmp/MyTemplate1.jar -outFile verify.out -outXML verify.xml -outSummary verify.summary

3. Verify the system, specifying additional Java properties that will be passed along to the ARE core JVM.

/QIBM/ProdData/OS/OSGi/healthcheck/bin/runARE.sh -template /tmp/MyTemplate1.jar -outFile verify.out -property are.log.level=FINE -property my.prop.key=someValue -property another.prop=anotherValue

4. Verify the system, useing collector mode. All the generated files will be packaged into the areReport.zip file, including report files, log files, etc.

   /QIBM/ProdData/OS/OSGi/healthcheck/bin/runARE.sh -template /tmp/MyTemplate1.jar -zipFile /my/dir/areReport.zip

5. Display the version of the ARE core on the current system.

   /QIBM/ProdData/OS/OSGi/healthcheck/bin/runARE.sh -version

arePlugin.sh

This script is the interface to start the ARE core for the sole purpose of running a single plugin. The primary use of this script is for unit testing a custom plugin prior to integrating it into a deployment template.

Location

This script is located in /QIBM/ProdData/OS/OSGi/healthcheck/bin/

Syntax

The command syntax is as follows (only required parameters are shown):

```
arePlugin.sh  -pluginName plugin_name  -elementPath element_path
```

Parameters

- **-pluginName *plugin_name*** – The full package and class name of the plugin to run. The full name must use the same naming convention as in the Java programming language, which means using dots, not slashes, as the package name separator. For example, if your plugin was in the com.ibm.are.sample package, and your plugin name was PluginTest, then the *plugin_name* parameter value would be com.ibm.are.sample.PluginTest. **This is a required parameter**. If the specified plugin name cannot be found, a com.ibm.are.core.AreLaunchException exception is thrown by the ARE core.
- **-elementPath *element_path*** – Any additional jars that need to be added to the element path can be added using this property. The path should be in the same format as a Java classpath, with a single colon used to separate path entries. Because a plugin will be packaged in a jar that is not included in the default element path, **this is a required parameter**.
- **-property *prop_key_and_value*** – Specifies a Java property that should be passed to the JVM when the ARE core is started. This parameter can be repeated as many times as needed, allowing multiple properties to be passed to the JVM when the ARE core is started. Note that the *prop_key_and_value* must be the Java property key and value pair, separated by an equal sign. This is an optional parameter.

  Examples:

  -property my.prop.key=propValue

  -property are.log.level=FINE

By default, all output from running the plugin is send to standard out, and no report output files are generated. If you would like the ARE core to generate report output files, just as it does when verifying a system using a template, the same command line parameters (-outFile, -outXML, -outSummary) can be specified. For details about how to use these parameters, see the Parameters documentation for the runARE.sh script.

Usage Examples

For all of the examples below, we assume that the plugin class name is com.ibm.are.example.SamplePlugin, and the sample plugin is packaged into a jar file named sample.jar located in the /tmp directory.

1. Run the SamplePlugin plugin

   /QIBM/ProdData/OS/OSGi/healthcheck/bin/arePlugin.sh -pluginName com.ibm.are.example.SamplePlugin -elementPath /tmp/sample.jar

areService.sh

This script is the interface to start the ARE core for the sole purpose of running a single service. The primary use of this script is for unit testing a custom service prior to integrating it into a deployment template.

Location

This script is located in /QIBM/ProdData/OS/OSGi/healthcheck/bin/

Syntax

The command syntax is as follows (only required parameters are shown):

```
areService.sh  -serviceName service_name  -elementPath element_path
```

Parameters

- **-serviceName *service_name*** – The full package and class name of the service to run. The full name must use the same naming convention as in the Java programming language, i.e. dots not slashes. So if your service was in the com.ibm.are.sample package, and your service name was ServiceTest, then the *service_name* parameter value would be com.ibm.are.sample.ServiceTest. **This is a required parameter**. If the specified service name cannot be found, a com.ibm.are.core.AreLaunchException exception is thrown by the ARE core.
- **-elementPath *element_path*** – Any additional jars that need to be added to the element path can be added using this property. The path should be in the same format as a Java classpath, with a single colon used to separate path entries. Because a service will be packaged in a jar that is not included in the default element path, **this is a required parameter**.
- **-property *prop_key_and_value*** – Specifies a Java property that should be passed to the JVM when the ARE core is started. This parameter can be repeated as many times as needed, allowing multiple properties to be passed to the JVM when the ARE core is started. Note that the *prop_key_and_value* must be the Java property key and value pair, separated by an equal sign. This is an optional parameter.

  Examples:

  -property my.prop.key=propValue

  -property are.log.level=FINE

By default, all output from running the service is send to standard out, and no report output files are generated. If you would like the ARE core to generate report output files, just as it does when verifying a system using a template, the same command line parameters (-outFile, -outXML, -outSummary) can be specified. For details about how to use these parameters, see the Parameters documentation for the runARE.sh script.

Usage Examples

For all of the examples below, we assume that the service class name is com.ibm.are.example.SampleService, and the sample service is packaged into a jar file named sample.jar located in the /tmp directory.

1. Run the SampleService service

   /QIBM/ProdData/OS/OSGi/healthcheck/bin/areService.sh -serviceName com.ibm.are.example.SampleService -elementPath /tmp/sample.jar

areFix.sh

This script is used to fix problems found during verification. Note that this script will only fix problems that can be automatically fixed, which is very likely only a subset of all problems found during verification. The script will invoke the ARE core and provide it with the necessary information to fix any problem that can be automatically fixed. If the verification included custom plugins that identified problems, and provided a description of how to fix the problems, this script will also be able to fix those problems. To a user of this script, the end result of invoking this script is that any problems identified during verification that can be automatically fixed will be fixed by running this script. By default, the script prompts the user to interactively acknowledge that it is ok to fix each problem before it is fixed. However, there is an option that forces the script to fix all problems automatically without asking for any input from the user.

For more information about automatically fixing problems, including detailed information about what problems can be automatically fixed, see the How to Automatically Fix Problems document on the ARE product Web site:

http://www.ibm.com/systems/power/software/i/are/documentation.html

Location

This script is located in /QIBM/ProdData/OS/OSGi/healthcheck/bin/

Syntax

The command syntax is as follows (only required parameters are shown):

```
areFix.sh  -report xml_report_filename
```

Parameters

- **-report *xml_report_filename*** – The name of the XML report generated by the ARE core as a result of verifying a system using a provided template. This report is generated as a result of verifying a system using the runARE.sh script or the ARE console. The report file name must be the fully qualified path to the XML report file. This report is utilized by the script and supporting Java code to determine what problems were found during verification, as well as what actions should be taken to fix the reported problems. **This is a required parameter**.

  Note that the XML report file name is, by default, generated by the ARE core based on the output file name specified for the -outFile parameter. The name

of the XML report file can be explicitly set using the -outXML parameter for the runARE.sh script.

- **-silent** – This parameter has no value. If specified, the script will run to completion without sending any information to standard output, or without prompting the user for any input. This is an optional parameter, and if it is not specified, the script will by default prompt the user to interactively acknowledge that it is ok to fix various problems before the fix action is done.

  IMPORTANT: If this parameter is specified, all problems that can possibly be fixed, as specified in the XML report, will be fixed automatically with no prompt or input from the user. When this parameter is specified, there is no opportunity to choose what problems to fix, or to safely stop the fixing of problems before the script completes its processing.

- **-elementPath *element_path*** – Specifies any additional jar files that the ARE core will need to use in order to fix detected problems. This parameter is only necessary if the verification included custom plugins that identified problems, and provided a description of how to fix the problems. In that case, it is typically necessary that the element path include the original template jar file, so that the ARE core has access to all of the Java classes included in the template. The element path should be in the same format as a Java classpath, with a single colon used to separate path entries. This is an optional parameter.

- **-property *prop_key_and_value*** – Specifies a Java property that should be passed to the JVM when the ARE core is started. This parameter can be repeated as many times as needed, allowing multiple properties to be passed to the JVM when the ARE core is started. Note that the *prop_key_and_value* must be the Java property key and value pair, separated by an equal sign. This is an optional parameter.

  Examples:

  -property my.prop.key=propValue

  -property are.log.level=FINE

Usage Examples

For all of the examples below, we will assume that the runARE.sh script has already been run using the following command:

/QIBM/ProdData/OS/OSGi/healthcheck/bin/runARE.sh -template /tmp/MyTemplate1.jar -outFile verify.out

This will generate three report files:

- verify.out
- verify.out.xml
- verify.out.summary.xml

For the purposes of automatically fixing problems, you are only interested in the XML report: verify.out.xml

1. Automatically fix detected problems that can be fixed using interactive, menu driven prompts.

   /QIBM/ProdData/OS/OSGi/healthcheck/bin/areFix.sh -report verify.out.xml

2. Automatically fix detected problems with no further input; all problems that can be fixed are fixed without any further action from the user.

   /QIBM/ProdData/OS/OSGi/healthcheck/bin/areFix.sh -silent -report verify.out.xml

3. Automatically fix detected problems with no further input; all problems that can be fixed are fixed without any further action from the user. Also re-direct all output to a set of report files. This is probably the most common way that the script is utilized.

   /QIBM/ProdData/OS/OSGi/healthcheck/bin/areFix.sh -silent -report verify.out.xml -outFile fix.out

   This will cause the following report output files to be generated:

   - fix.out
   - fix.out.xml
   - fix.out.summary.txt

4. Automatically fix detected problems, some of which were identified, and fixes provided by, one or more custom plugins. This example assumes that the custom plugins are contained in the /tmp/MyTemplate1.jar JAR file.

   /QIBM/ProdData/OS/OSGi/healthcheck/bin/areFix.sh -report verify.out.xml -outFile fix.out -elementPath /tmp/MyTemplate1.jar

runAreLauncher.sh

This script is only used by other scripts. It implements some common components and logic that is necessary to invoke the ARE core.

areCommonCode.sh

This script is only used by other scripts. It contains several “common code” script functions, such as setting the JAVA_HOME environment variable.