



AT&T Developer Program

Application Resource Optimization

Adding Custom Best Practices in Open Source ARO

Revision Date **December 28, 2012**



Legal Disclaimer

This document and the information contained herein (collectively, the "**Information**") is provided to you (both the individual receiving this document and any legal entity on behalf of which such individual is acting) ("**You**" and "**Your**") by AT&T, on behalf of itself and its affiliates ("**AT&T**") for informational purposes only. AT&T is providing the Information to You because AT&T believes the Information may be useful to You. The Information is provided to You solely on the basis that You will be responsible for making Your own assessments of the Information and are advised to verify all representations, statements and information before using or relying upon any of the Information. Although AT&T has exercised reasonable care in providing the Information to You, AT&T does not warrant the accuracy of the Information and is not responsible for any damages arising from Your use of or reliance upon the Information. You further understand and agree that AT&T in no way represents, and You in no way rely on a belief, that AT&T is providing the Information in accordance with any standard or service (routine, customary or otherwise) related to the consulting, services, hardware or software industries.

AT&T DOES NOT WARRANT THAT THE INFORMATION IS ERROR-FREE. AT&T IS PROVIDING THE INFORMATION TO YOU "AS IS" AND "WITH ALL FAULTS." AT&T DOES NOT WARRANT, BY VIRTUE OF THIS DOCUMENT, OR BY ANY COURSE OF PERFORMANCE, COURSE OF DEALING, USAGE OF TRADE OR ANY COLLATERAL DOCUMENT HEREUNDER OR OTHERWISE, AND HEREBY EXPRESSLY DISCLAIMS, ANY REPRESENTATION OR WARRANTY OF ANY KIND WITH RESPECT TO THE INFORMATION, INCLUDING, WITHOUT LIMITATION, ANY REPRESENTATION OR WARRANTY OF DESIGN, PERFORMANCE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, OR ANY REPRESENTATION OR WARRANTY THAT THE INFORMATION IS APPLICABLE TO OR INTEROPERABLE WITH ANY SYSTEM, DATA, HARDWARE OR SOFTWARE OF ANY KIND. AT&T DISCLAIMS AND IN NO EVENT SHALL BE LIABLE FOR ANY LOSSES OR DAMAGES OF ANY KIND, WHETHER DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL, PUNITIVE, SPECIAL OR EXEMPLARY, INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, LOSS OF GOODWILL, COVER, TORTIOUS CONDUCT OR OTHER PECUNIARY LOSS, ARISING OUT OF OR IN ANY WAY RELATED TO THE PROVISION, NON-PROVISION, USE OR NON-USE OF THE INFORMATION, EVEN IF AT&T HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH LOSSES OR DAMAGES.



Table of Contents

1. Introduction	4
2. Best Practice Screen Layout	5
2.1 Tests Conducted Section.....	7
2.2 The Detailed Results Section	9
3. Best Practice Development	11
4. Creating a new Best Practice Group	12
5. Creating a new Best Practice Test	13
6. Adding Custom Best Practices to the ARO UI	15



1. Introduction

The Application Resource Optimizer (ARO) has two components that help you analyze and improve your application.

The ARO Data Collector is the component of ARO that captures the data traffic of a mobile device and stores that information in trace files that can be analyzed using the ARO Data Analyzer. The ARO Data Analyzer evaluates the trace data collected from an application and generates statistical and analytical results based on recommended best practices.

The best practice tests that are included in the ARO Data Analyzer are designed to evaluate the most common causes and effects of energy and data inefficiency in mobile applications. However, the ARO Data Analyzer was designed to be extensible with the knowledge that these best practices may not be the only criteria you want to evaluate your application against.

With the open source code for ARO available at <https://github.com/attdevsupport/ARO>, you're able to build your own version of ARO with any new or modified best practice tests that you wish to include.

This document describes how the best practice tests are organized within the ARO Data Analyzer, looks at the classes and methods involved in adding and displaying best practices, and describes how to add a custom best practices.



2. Best Practice Screen Layout

The Best Practices/Results tab (shown in the following figure) is the first tab in the ARO Data Analyzer, it displays the results for all of the Best Practices tests that are conducted on the loaded trace files.

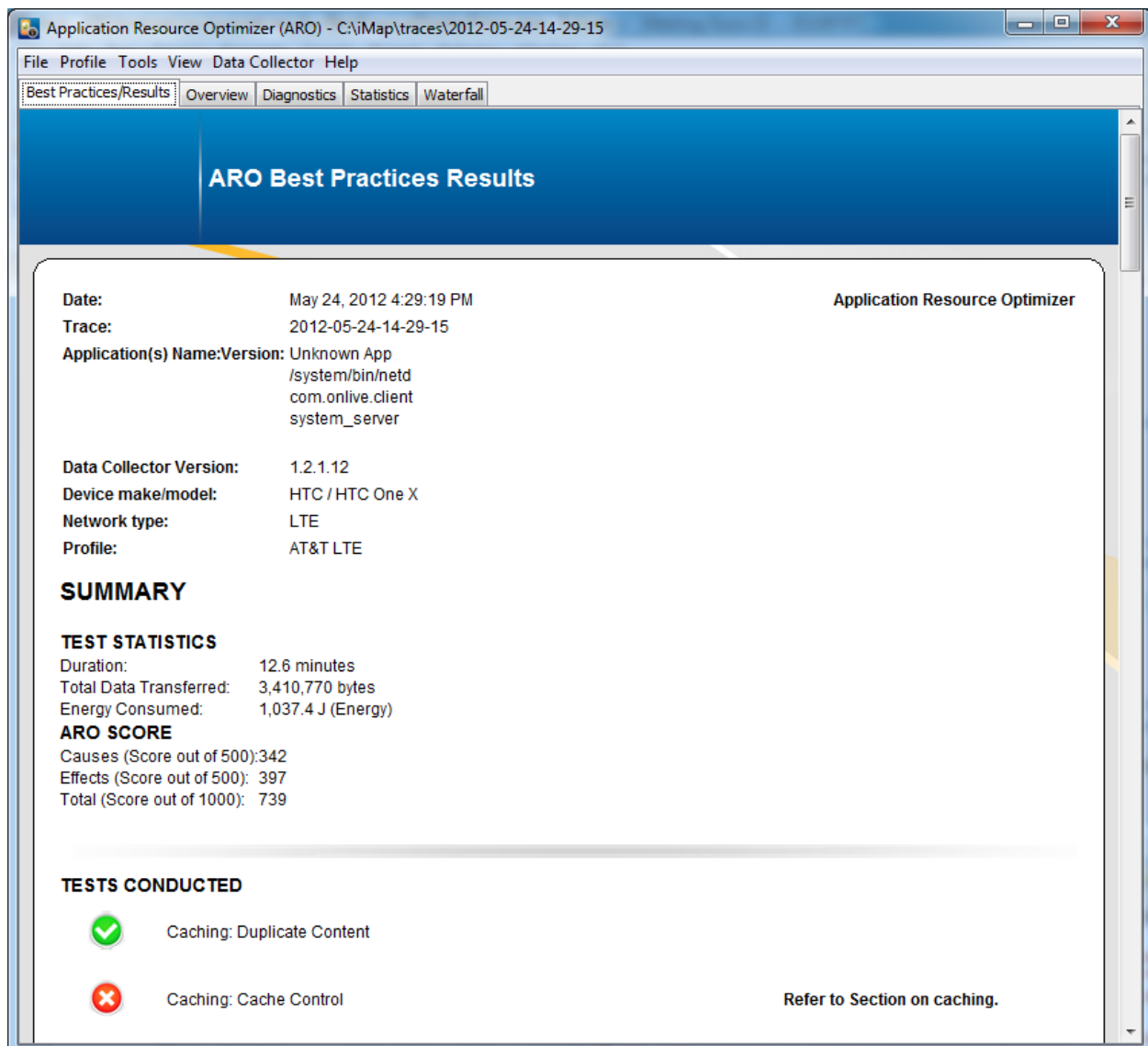


Figure 1 – The ARO Best Practices/Results Tab



When a trace file is loaded into the ARO Data Analyzer, the Best Practices / Results tab displays the following sections:

- A Header section that contains information about the device, the applications that were tested, the network, and the profile type that were used when the trace was captured.
- A Summary section that contains high-level statistics about the trace, including a Trace Score that quantifies the test results by measuring how the application mitigated basic causes and limited the effects of data and energy inefficiency.
- A Tests Conducted section that contains high-level (pass/fail) test results.
- A detailed results section that contains blocks of results that group the Best Practices tests into three test categories; Using Cache & Cache Problems, Managing Connections, and Others.

The two sections of the Best Practices / Results tab that must be modified when a new best practice test is added are the Test Conducted section and the detailed results sections that follow it.



2.1 Tests Conducted Section

The pass/fail results of the individual best practice tests are shown in the Tests Conducted section. In this section (shown in the following figure), each best practice test that is run during trace analysis is listed.

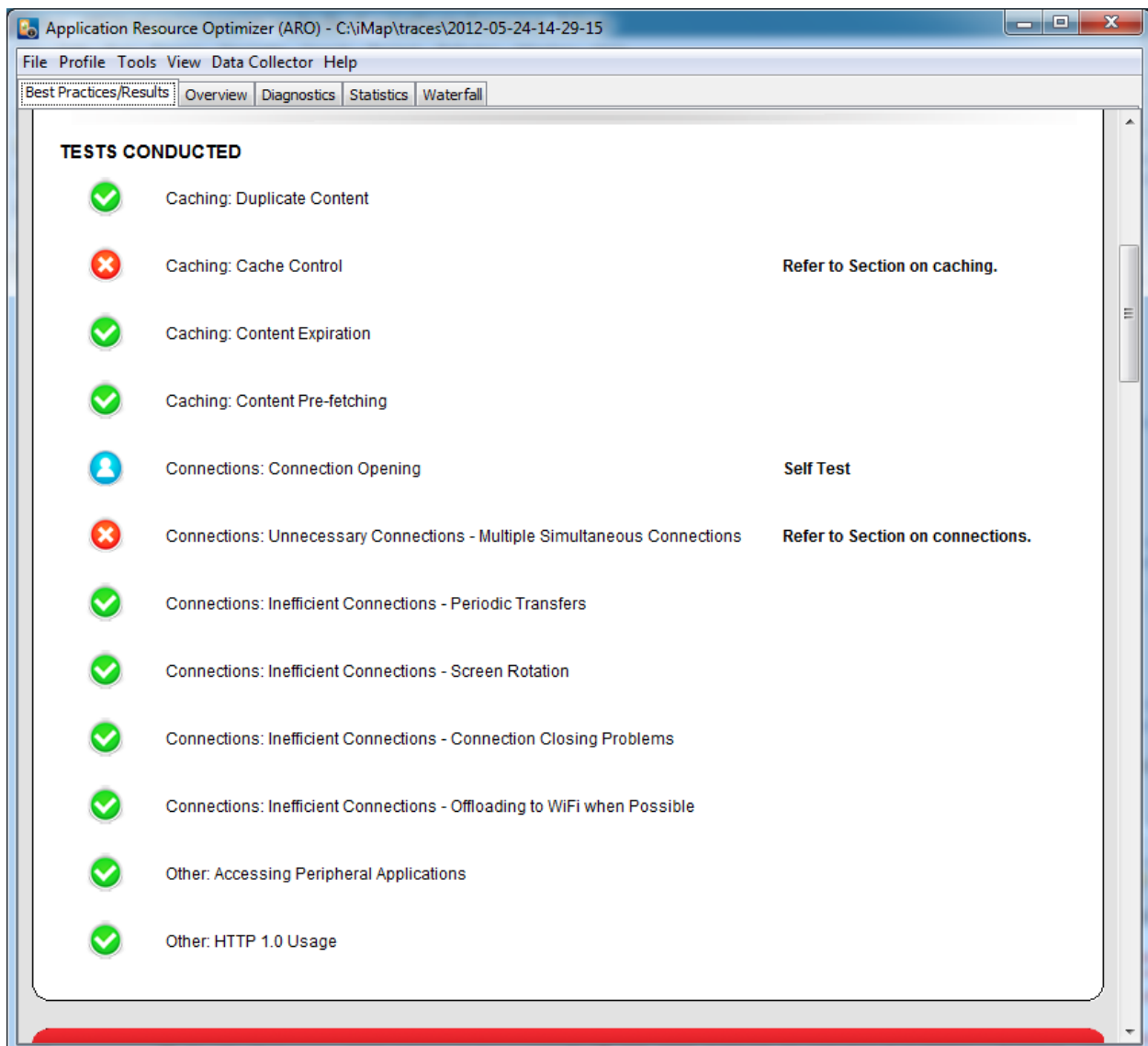





Figure 2 – The Tests Conducted section of the Best Practices/Results Tab.



For each best practice test, an icon is displayed indicating the status of the test: Pass, Fail, or Manual. The following chart shows these icons and the test status that they represent:

Icon	Color	Description
	Green	The best practice test has passed.
	Red	The best practice test has failed.
	Blue	The best practice test is a manual test.

A *manual test* is a test that is not run automatically with pass/fail criteria by the ARO Data Analyzer. It is a guideline for informing a developer of a best practice and giving the developer guidance on observations that can be made about an application. A “Self Test” link is displayed next to each manual test to allow the developer to see more detailed information about the best practice.

For each failed test, a referral link is displayed which provides access to the detailed results section for that test.



2.2 The Detailed Results Section

After the Tests Conducted section, subsequent blocks on the Best Practices/Results tab contain detailed information about the best practice tests. Each block contains a logical group of best practices. The following figure shows the “Using Cache & Caching Problems” group.

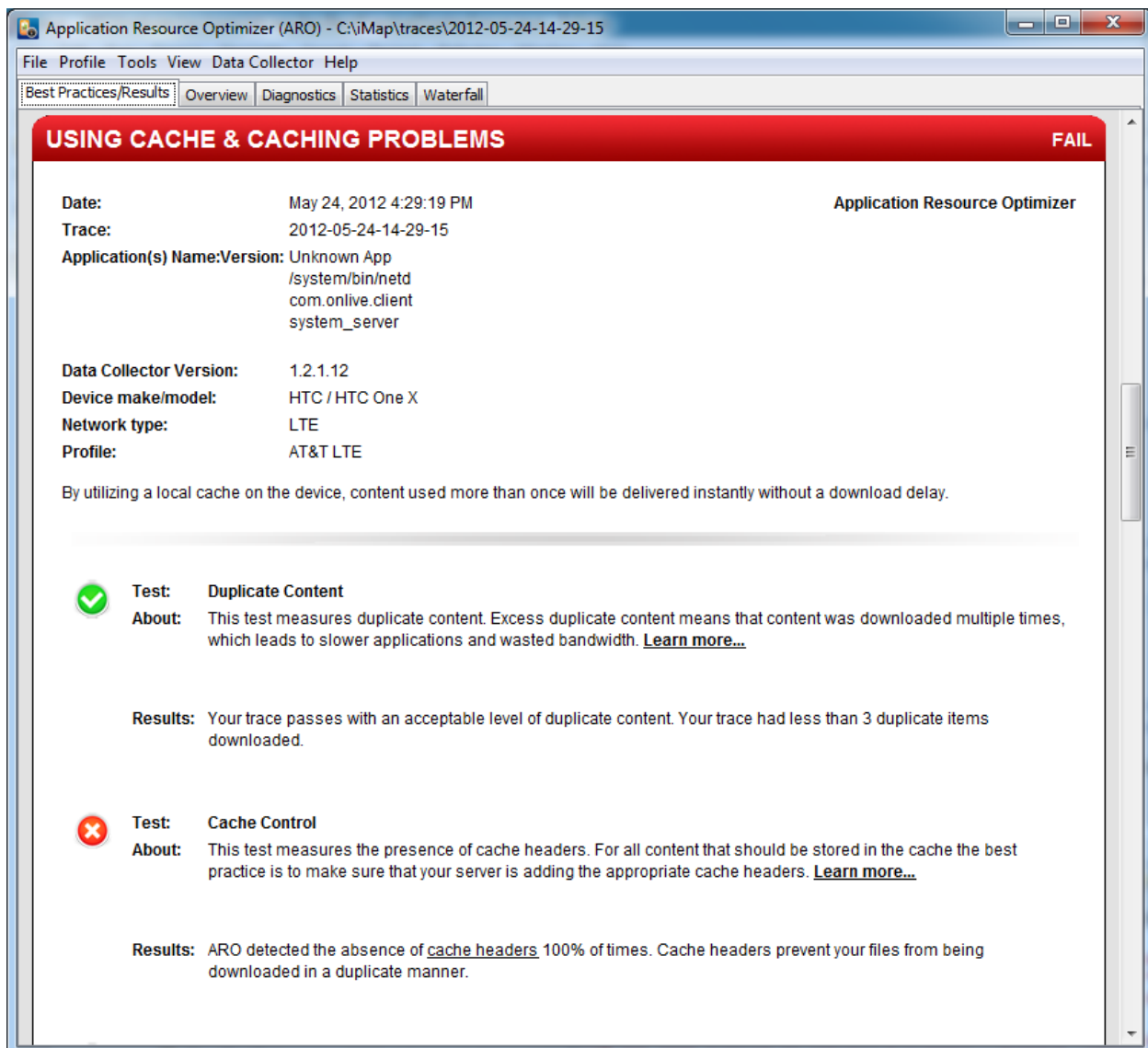


Figure 3 – The Using Cache & Caching Problems group



Each group of best practices has a title bar that indicates the status of the best practice tests within the group. *If any test within the group fails, the title bar will be red to indicate the failure. If all of the tests within the group pass, the title bar will be green.*

After the title bar, the information from the Header Section is repeated and is followed by a description of the best practice group.

After the group description, detailed information is displayed for each best practice, including a pass/fail/manual icon, the test name, a description of the test, and a text description of the results based upon the current trace analysis.

When the user clicks on the results text of a failed best practice, they are directed to the part of the trace information that triggered the failure. For example, when you click on the detailed results for a failed Duplicate Content test, you will be directed to the Duplicate Content table of the “Overview” tab for further details.



3. Best Practice Development

The ARO Open Source code contains interfaces, classes, and methods that are designed to let you customize the existing set of ARO Best Practices by adding or modifying Best Practices Groups, and the Best Practices tests within the groups.

The interfaces and classes contained in the **com.att.aro.bp** ARO Java package are used to define how the best practice tests are displayed on the ARO Best Practices tab.

By creating a class that implements the **BestPracticeDisplay** interface, the display for a new best practice test can be added to the Best Practices tab in the ARO Data Analyzer.

To add a new block of tests to the tab, the **BestPracticesDisplayGroup** interface should be implemented along the lines of the default implementation in the **BestPracticesDisplayGroupImpl** class. The new best practice groups and tests are added to the tab by modifying or extending the **BestPracticeDisplayFactory** class.

The following table describes the interfaces and classes in the **com.att.aro.bp** package that are used to display Best Practices in the ARO Data Analyzer:

Class or Interface	Description
BestPracticeDisplay	Defines the interface for displaying a best practice test. New best practice test classes are created by implementing this interface.
BestPracticeDisplayFactory	A singleton factory class that creates the default set of best practices to be displayed in the ARO Data Analyzer. This class can be modified or extended to allow for the addition of custom best practices and best practice groups.
BestPracticeDisplayGroup	Defines the interface for displaying a group (or block) of best practice tests on the Best Practices tab of the ARO Data Analyzer.
BestPracticeDisplayGroupImpl	The default implementation of the BestPracticeDisplayGroup interface. This class represents the display of a group of best practice tests.
BestPracticeExport	A class that defines additional data to be exported for a best practice test as part of the feature in the ARO Data Analyzer that exports data to a .csv file.



4. Creating a new Best Practice Group

To create a new block of best practice details for the Best Practices/Results tab, a new implementation of the **BestPracticeDisplayGroup** interface or a new instance of the **BestPracticeDisplayGroupImpl** class must be created to define the properties for the group.

The following table describes the methods in the **BestPracticeDisplayGroup** interface that return the properties of a best practices group.

Method	Description
String getHeaderName()	Returns the text to be displayed in the block header for the best practice group.
String getDescription()	Returns the description of the best practice group. This description is displayed in the header of the best practice group block.
String getReferSectionName()	Returns a short name of the best practice group that will be used to refer to the section in a sentence such as: "Refer to the section on <i><referSectionName></i> ".
Collection<BestPracticeDisplay> getBestPractices()	Returns a collection of the best practices that are included in this group.



5. Creating a new Best Practice Test

To create a new best practice test, a developer must create a new implementation of the **BestPracticeDisplay** interface and include it in a best practice group. This interface encapsulates all of the features of the best practice. The following table describes the methods in the **BestPracticeDisplay** interface.

Method	Description
String getOverviewTitle()	Returns the title of the best practice test displayed in the Tests Conducted list on the Best Practices tab.
String getDetailTitle()	Returns the title of the best practice displayed in the detail panel for the best practices group.
boolean isSelfTest()	Indicates whether this best practice is a manual test.
String getAboutText()	Returns the detailed description of the best practice test that is displayed with the label "About:" in the detail panel for the best practice group.
URI getLearnMoreURI()	Returns a URI that is a link to more detailed information about the best practice. If this property is not null, the URI is displayed as a "Learn More" link in the detail panel.
boolean isPass(TraceData.Analysis analysis)	Returns a value that indicates whether the specified best practices test has passed. If the best practice is a "self-test", this method is not called.
String resultText(TraceData.Analysis analysisData)	<p>Returns a string containing the results of the best practices test based on the specified analysis data. This text is displayed in the results text field of the best practice details.</p> <p>If the test has failed, this message may contain hyperlinks that can be used to display more detail about the failed test. See the performAction method.</p>



void performAction(javax.swing.event.HyperlinkEvent h, ApplicationResourceOptimizer parent)	Performs an action when the user clicks on a hyperlink in the results of a failed test. This method allows the developer to display more detail about a failed test or to perform some action within the application based on the result of the test. If the best practice is a “self-test”, this method is not called.
List<BestPracticeExport> getExportData(TraceData.Analysis analysisData)	Returns a List of BestPracticeExport objects containing the additional data that is included when best practice information is exported to a .csv file. This method allows a developer to include extra information, in addition to the best practice title and the result of the test (pass, fail, or self-test), when exporting best practice information.



6. Adding Custom Best Practices to the ARO UI

The **BestPracticeDisplay** interface encapsulates all of the features of a best practice. To create a new best practice test, an implementation of the **BestPracticeDisplay** interface must be created and included in a best practice group.

The display of best practices information in the ARO Data Analyzer is driven by the **BestPracticeDisplayFactory** class. Developers can customize the display of pre-defined or custom best practice groups on the Best Practices/Results tab by modifying or extending this factory class.

The following table describes the methods in the **BestPracticeDisplayFactory** class.

Method	Description
BestPracticeDisplayFactory.getInstance()	This static method is called by the ARO UI to get an instance of the BestPracticeDisplayFactory that it uses to retrieve the best practice display configuration. This method can be modified to return a custom best practice factory.
Collection<BestPracticeDisplayGroup> getBestPracticeDisplay()	The ARO UI calls this method on the BestPracticeDisplayFactory to get the best practice configuration. By default, the getBestPracticeDisplay method returns the pre-defined ARO best practice configuration. This method can be overridden or modified to provide a custom best practice configuration.