Introduction to the WebSphere Cast Iron Management API

Paul Ray

March 07, 2012

This article describes the WebSphere Cast Iron Management API and shows you how to use it manage and monitor a WebSphere Cast Iron environment. The article also shows you how to configure an orchestration to automate the export of a Cast Iron repository, without having to write the code that would be required without the Management API.

Introduction

If you rely on IBM® WebSphere® WebSphere Cast Iron as a strategic platform for your business-critical cloud integrations, then you know how important it is to effectively manage and monitor the health of your appliances and orchestrations. For many Cast Iron installations, the combination of the Web Management Console (WMC) and Command Line Interface (CLI) management tools are sufficient to get the job done. But when you need more flexibility in how you manage and monitor Cast Iron, it’s time to look at your third option: the WebSphere Cast Iron Management API.

The Management API gives you the freedom to customize the user experience of managing and monitoring your appliances in order to streamline your operations. A task that may take several manual steps to accomplish in the WMC for example, can be automated with a custom user interface that reduces or eliminates these steps entirely. This custom interface can take several forms: a command-line script, a Web page, or a Cast Iron orchestration. In this article, you’ll see an example of an orchestration that consumes the Management API to fully automate the task of exporting a Cast Iron repository.

Benefits of the Management API are not limited to IT experts. For business users, it can play a key role in surfacing appliance information within mashups and other variations of "integration on the glass," and provide insight into such things as the path a sales order took from one system to the next.

Overview: The Management API and its services

The Management API is implemented as a set of web services hosted in the Cast Iron offerings. These services are provided on all available form factors:

- WebSphere Cast Iron Live
- WebSphere Cast Iron Hypervisor Edition
• WebSphere DataPower Cast Iron Appliance XH40

Some elements of the Management API are not applicable to all form factors. For example, the cloud offering does not include a staging database, so the methods and properties of the Staging service apply only to the physical and virtual appliances.

Web Services Definition Language (WSDL) descriptions of these services are provided via URLs on an appliance, and they can be downloaded and converted to stubs. Client scripts and applications can interact with these stubs to consume the services via SOAP messages over HTTP or HTTPS. If you've worked with WSDLs and SOAP-based web services before, then wiring up to the Management API should be straightforward. The WSDLs that describe the Management API, and the types of operations they provide, include:

**Security**
- Users, groups, roles, and permissions

**Staging**
- Staging database

**Deployment**
- Projects, rules, and configurations

**System**
- Network configuration and execution of commands

**OrcMon**
- Job logs

**LogNotif**
- Notifications and the system log

The **Security** WSDL for example, exposes a `disableUser()` method that disables a user from authenticating with the appliance.

The WSDLs are accessible via the management port that is configured for your appliance. This port is the same one that you use when accessing the WMC or CLI. Thus, if your management network is configured as `ci-mgmt.mydomain.com` for example, the **Security** WSDL will be located at `http(s)://ci-mgmt.mydomain.com/ws/security?wsdl`.

For complete information on the WSDLs, see the [WebSphere DataPower Cast Iron Management API Guide](https://www.ibm.com/docs/en/cic-xh40).

Armed with this comprehensive set of WSDLs exposed by the Management API, you are limited only by your imagination as to the types of solutions you can build in order to streamline your appliance management operations or get better insight into your Cast Iron environment. Below is a list of sample use cases that demonstrates the types of tasks that can be automated and the clients that can be used to consume the Management API's services, and should help spawn your own ideas:

**Scheduled Downtime Rules Configuration**
- You can write an ASP.NET page to schedule downtimes of one or more orchestrations from a single screen.
User Accounts Inactivation
You can inactivate a user account using a script as part of an established account deactivation procedure.

Windows PowerShell Command Execution
You can write scripts and cmdlets that enable you to manage an appliance from within the Windows PowerShell console.

System Log Portlet or RSS Feed
You can surface information directly from your system log to a portlet in a portal or any RSS feed consumer.

Orchestration for More Intelligent Notification Routing
You can build an orchestration that scans the system log for particular events and sends e-mail notification to select recipients, augmenting standard notification policies.

Mobile Web Application
You can build a mobile web application that enables you to perform common appliance management tasks from your mobile device.

Generally speaking, as long as you are using a client or development language that can consume SOAP-based web services, you can manage your Cast Iron environment using the Management API. Since WebSphere Cast Iron supports web services as an endpoint, you can also call these services from an orchestration via point-and-click configuration, which will be demonstrated below in building an orchestration to back up your Cast Iron repository.

Working with the Management API

The first step in working with the Management API is to download the WSDL(s) that you plan to consume. You can then use these files to generate the stubs that your code interacts with. At a minimum, you need to download the Security WSDL, because it contains the description of the login method that you use to establish a Management API session. When you establish a session, the login method returns a session ID, and this unique 32-character string is passed as a SOAP header in all subsequent calls to the Management API.

To download the Security WSDL, point your Web browser to its location on your appliance at http(s)://<appliance-management-port>/ws/security?wsdl and save the XML-encoded response to a file.

After you've downloaded the WSDLs you need to work with, follow the procedure dictated by the development client or language you are using in order to create the stubs for the services. Using Microsoft Visual Studio for example, you would add a Web Reference. In Cast Iron Studio, you would import the WSDL.

After your WSDLs are imported, you are ready to code (or configure, in the case of a Cast Iron orchestration). Since you need a session ID if you want to do anything useful with the Management API, call the login method of the Security WSDL, passing it the username and password of a Cast Iron account with appropriate access for the task that you want to accomplish. If you were to trace the SOAP request sent to the Management API to log in and establish a session, the payload may look like this:
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <soap:Body>
  <login xmlns="http://www.approuter.com/schemas/2008/1/security">
   <username>admin</username>
   <password>mypassword</password>
  </login>
 </soap:Body>
</soap:Envelope>

On return from the appliance, a trace of the SOAP response payload would show a session ID being returned:

<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
   xmlns:ns2="http://www.approuter.com/schema/router/1000/security/03">
   <ns3:sessionId>87E0F9FF84DC54675D0B145185033576</ns3:sessionId>
  </ns3:loginResponse>
 </S:Body>
</S:Envelope>

How you set the sessionId SOAP header in subsequent calls to the Management API depends on the client's implementation of SOAP. Suppose you are building a Windows console application using the Microsoft .NET framework that consumes the Management API to get a list of users on an appliance. You've created a Web Reference in your Visual Studio project using the downloaded WSDL, and wired up the call to login to return the session ID to a string variable. To pass this variable in a SOAP header to the listUsers method, you copy it to the Text property of the sessionId type (defined as @string, a subclass of SoapHeader in the System.Web.Services.Protocols namespace). The sample code below demonstrates this technique:

```csharp
using System;
using MgmtApi.Security;

namespace MgmtApi
{
    class Program
    {
        static void Main(string[] args)
        {
            // establish Management API session
            // note: additional code may be needed here if HTTPS is enforced
            SecurityManager securityManager = new SecurityManager();
            string loginResult = securityManager.login(args[0], args[1]);

            // set the sessionId SOAP header
            Security.@string sessionIdSoapHeader = new Security.@string();
            sessionIdSoapHeader.Text = new string[] { loginResult }; 
            securityManager.sessionId = sessionIdSoapHeader;

            // write list of users
            foreach (string user in securityManager.listUsers(new listUsers()))
            {
                Console.WriteLine(user);
            }
        }
    }
}
The call to `listUsers` takes a new instance of `listUsers` as a parameter. It is this parameter that contains the information needed to construct the body of the SOAP message.

Putting a trace on the request to `listUsers` would show a payload like this, with the session ID in the SOAP header:

```xml
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Header>
    <sessionId xmlns="http://www.approuter.com/schemas/2008/1/security">
      87E0F9FF84DC54675D0B145185033576</sessionId>
  </soap:Header>
  <soap:Body>
    <listUsers xmlns="http://www.approuter.com/schemas/2008/1/security"/>
  </soap:Body>
</soap:Envelope>
```

The response SOAP payload from the call to `listUsers` then looks like this:

```xml
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:ns3="http://www.approuter.com/schemas/2008/1/security">
  <S:Body>
    <ns3:listUsersResponse xmlns:ns3="http://www.approuter.com/schemas/2008/1/security">
      <ns3:users>system</ns3:users>
      <ns3:users>admin</ns3:users>
    </ns3:listUsersResponse>
  </S:Body>
</S:Envelope>
```

Since Cast Iron supports web services as endpoints, you can import the Management API WSDLs into Cast Iron Studio and then call the services from an orchestration. However, since the `sessionId` header is imported as being optional, it does not appear by default when mapping inputs to these services. To pass a session ID to a Management API method in Cast Iron Studio (illustrated in Figure 1 below), follow these steps:

1. In Cast Iron Studio, open the Invoke Service activity pointing to the web service you are trying to invoke, and click on Map Inputs.
2. Right-click anywhere in the To Activity area of the screen and select Show Optional Parameters.
3. Expand `{optional} header` to locate the `sessionId` element.
4. Click Select Inputs and select the variable containing the session ID returned from the `login` call.
5. Map this variable to the `sessionId` element.
Building an orchestration to automate a repository export

The project repository in your appliance contains all your project configuration and user settings. Network configurations, users and groups, notification policies, downtime rules, and other settings are persisted in this repository. It can be used to copy all of your projects from one appliance to another, if the need arises. In the event of a disaster, having a current copy of this repository enables you to get your appliance up and running on a cold standby appliance in a fraction of the time it would take to rebuild from scratch.

You can back up a repository manually using the WMC or CLI. But using the Management API, you can automate and schedule this task so that it is done automatically. That way, you will always be sure that your backup contains the latest versions of your configuration. Alternatively, you can use this same orchestration to perform an ad hoc repository export from another client interface, perhaps as part of a script that deploys orchestrations.

In this section, you'll learn how to build a simple orchestration that exports a repository on a scheduled or ad hoc basis and sends it to a secure FTP server using a configuration and no code approach. You can download the Cast Iron Studio project used in this article at the bottom of the article. It was built using WebSphere Cast Iron Studio V6.0.0.4.

Before diving into the details, it's important to point out that importing and exporting a repository is not supported in WebSphere Cast Iron Live, since projects, logs, users, and other persistent data are backed up automatically in that environment. However, you can still use the technique described here to establish an API session and pass session information to subsequent API calls in an orchestration.

Solution overview

The solution, shown below in Figure 2, consists of a single Cast Iron orchestration with the following process flow:

1. The Pick activity executes, kicking off the orchestration either on a schedule or via an ad hoc HTTP request.

Figure 1. Mapping session ID to SOAP header in Cast Iron Studio

This configuration task is a critical step in making the sample orchestration in the next section work.
2. A web service Invoke activity calls the `login` method of the Security service to establish an API session.

3. Using the session ID, a web service Invoke activity calls the `exportRepository` method of the Deployment service to export the repository.

4. An FTP `Put File` activity copies the exported repository to an FTP server.

5. A web service Invoke activity calls the `logout` method of the Security service to end the API session.

A CatchAll activity catches any exceptions that are raised and logs the fault message (with a level of CRITICAL) to the System Log via a Log Message activity. In a production scenario, you would ideally have a common error handler process any exceptions, but to keep things simple, it's a simple logging action.

**Figure 2. ExportRepository orchestration as viewed in WebSphere Cast Iron Studio**

Building the solution

1. Open Cast Iron Studio and create a new project.
2. Create a set of Configuration Properties for the project. See Table 1 for a list of recommended properties.
3. Download the Management API Security WSDL and save to a file.
4. Download the Management API Deployment WSDL and save to a file.
5. Import the Security WSDL file into Cast Iron Studio.
6. Import the Deployment WSDL file into Cast Iron Studio.
7. Add and configure the following endpoints, using the Configuration Properties defined previously where applicable:
   - HTTP endpoint with **Integration Appliance Receives Request** selected
   - Web service endpoint for Security WSDL
   - Web service endpoint for Deployment WSDL
   - FTP endpoint
8. Using the layout of the orchestration in Figure 2 above as a guide, drag the following activities onto the canvas of the orchestration:
   - Pick logic activity that contains both **Scheduled Job** utility and **Receive Request** HTTP activities
• Invoke Service web service activity with the Security web service as endpoint, and login as WSDL Operation
• Invoke Service web service activity with the Deployment web service as endpoint, and exportRepository as WSDL Operation
• Invoke Service web service activity with the Security web service as endpoint, and logout as WSDL Operation
• Put File FTP activity pointing to FTP endpoint
• CatchAll logic activity
• Log Message utility activity

9. In the Invoke Service activity pointing to the login operation, set the login username and password elements in the body variable in the Map Inputs screen to the username and password defined in the Configuration Properties.
10. In the Map Outputs screen of that activity, map the sessionId element of the body variable from the activity to a new orchestration string variable. This contains the session ID returned from the login method and will be used to pass into subsequent API calls.
11. In Map Inputs screen of the Invoke Service activity pointing to the exportRepository operation, perform the technique described earlier to map the session ID returned from the login call to the sessionId header for the call to exportRepository.
12. In the Map Outputs screen of that activity, copy the response from the activity to a new orchestration variable via the Copy button. It contains the exported repository and will be used to copy to the FTP server.
13. In the Put File FTP activity, ensure that Binary is selected as the file type to be saved as.
14. In the Map Input screen of that activity, select the exported repository variable as input and map it to the data variable of the activity. Also, provide a file name for the filename variable.
15. In Map Input screen of the Invoke Service activity pointing to the logout operation, map the session ID to the sessionId header.

**Table 1. Configuration properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthPassword</td>
<td>Password</td>
<td>Password of user account logging in to Management API</td>
</tr>
<tr>
<td>AuthUsername</td>
<td>String</td>
<td>Username of user account logging in to Management API</td>
</tr>
<tr>
<td>DeploymentWSLocation</td>
<td>String</td>
<td>URL of the Deployment web service endpoint</td>
</tr>
<tr>
<td>FtpHostName</td>
<td>String</td>
<td>Host name of the FTP server</td>
</tr>
<tr>
<td>FtpHostPort</td>
<td>String</td>
<td>Port on host server that FTP service is listening on</td>
</tr>
<tr>
<td>FtpInDirectory</td>
<td>String</td>
<td>Target directory on FTP server that repository will be copied to</td>
</tr>
<tr>
<td>FtpPassword</td>
<td>Password</td>
<td>Password of the FTP user account</td>
</tr>
<tr>
<td>FtpUsername</td>
<td>String</td>
<td>Username of the FTP user account</td>
</tr>
<tr>
<td>HttpReceiverPort</td>
<td>String</td>
<td>Port on appliance to listen for HTTP requests on</td>
</tr>
<tr>
<td>HttpReceiverUrl</td>
<td>String</td>
<td>The path portion of the URL that triggers the orchestration to run</td>
</tr>
</tbody>
</table>
After successful verification of your orchestration in Cast Iron Studio, you are ready to publish it to your appliance and run it there. If you are forcing HTTPS on the management port of your appliance, you may experience a certificate error that prevents the orchestration from running. The next section describes a workaround for this error.

**Using the Management API with SSL**

If you've configured your appliance to enforce only HTTPS traffic to your management port, and you are invoking the Management API services from an orchestration running on the same appliance as in the solution above, your default appliance configuration may generate the following error, which you may not have seen while verifying the orchestration in Cast Iron Studio:

```
Exception while processing web service activity:
Error message: com.ibm.jsse2.util.g: PKIX path building failed:
java.security.cert.CertPathBuilderException:
unable to find valid certification path to requested target.
Caused by: com.ibm.jsse2.util.g: PKIX path building failed:
java.security.cert.CertPathBuilderException:
unable to find valid certification path to requested target.
```

If you see this error, navigate to the Security -- Certificates page in the WMC and ensure that VPeer is unchecked for the Client SSL setting. You'll need to either restart the appliance, or else undeploy and redeploy the orchestration causing the errors.

In those instances where you are not using Cast Iron as the Management API consumer, you may need to add logic to your client code to address certificate verification where HTTPS is required.

**Conclusion**

If you need a more flexible way to manage and monitor your Cast Iron appliances and orchestrations that goes beyond what the WMC and CLI interfaces provide, the Management API is the answer. By harnessing the standards-friendly services of the Management API, you can customize your user experience to increase ease-of-use and overall ROI. Once you are familiar with the services that the Management API offers and the basic requirements for interfacing with them, you are limited only by your imagination in what you can do with it.
## Downloadable resources

<table>
<thead>
<tr>
<th>Description</th>
<th>Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExportRepository sample orchestration</td>
<td>RepoBackup.zip</td>
<td>33 KB</td>
</tr>
</tbody>
</table>
Related topics

- **WebSphere Cast Iron resources**
  - **WebSphere DataPower Cast Iron Management API guide**
    Information on the Management API for the Cloud and Integration Appliances.
  - **WebSphere Cast Iron V6.1 information center**
  - **IBM Redbook: Getting started with IBM WebSphere Cast Iron Cloud Integration**
    Detailed introduction to the development and administrative interfaces for WebSphere Cast Iron.
  - **WebSphere Cast Iron Cloud Integration product page**
    Product descriptions, product news, training information, support information, and more.
  - **WebSphere Cast Iron Cloud Integration product library**
    Product announcements, case studies, white papers, and more.
  - **Cast Iron community forums**
    Get answers to your technical questions and share your expertise with other WebSphere Cast Iron users.
  - **Cast Iron skill building**
    Learn the "configuration not coding" approach to connecting your public and private clouds and on-premise applications, and develop administration and configuration skills for WebSphere Cast Iron.

- **WebSphere resources**
  - **developerWorks WebSphere developer resources**
    Technical information and resources for developers who use WebSphere products.
    developerWorks WebSphere provides product downloads, how-to information, support resources, and a free technical library of more than 2000 technical articles, tutorials, best practices, IBM Redbooks, and online product manuals.
  - **developerWorks WebSphere application integration developer resources**
    How-to articles, downloads, tutorials, education, product info, and other resources to help you build WebSphere application integration and business integration solutions.
  - **Most popular WebSphere trial downloads**
    No-charge trial downloads for key WebSphere products.
  - **WebSphere forums**
    Product-specific forums where you can get answers to your technical questions and share your expertise with other WebSphere users.
  - **WebSphere on-demand demos**
    Download and watch these self-running demos, and learn how WebSphere products and technologies can help your company respond to the rapidly changing and increasingly complex business environment.
  - **developerWorks WebSphere weekly newsletter**
    The developerWorks newsletter gives you the latest articles and information only on those topics that interest you. In addition to WebSphere, you can select from Java, Linux, Open source, Rational, SOA, web services, and other topics. Subscribe now and design your custom mailing.
  - **WebSphere-related books from IBM Press**
Convenient online ordering through Barnes & Noble.

- **WebSphere-related events**
  Conferences, trade shows, Webcasts, and other events around the world of interest to WebSphere developers.

- **developerWorks resources**
  - **Trial downloads for IBM software products**
    No-charge trial downloads for selected IBM® DB2®, Lotus®, Rational®, Tivoli®, and WebSphere® products.
  - **developerWorks blogs**
    Join a conversation with developerWorks users and authors, and IBM editors and developers.
  - **developerWorks cloud computing resources**
    Access the IBM or Amazon EC2 cloud, test an IBM cloud computing product in a sandbox, see demos of cloud computing products and services, read cloud articles, and access other cloud resources.
  - **developerWorks tech briefings**
    Free technical sessions by IBM experts to accelerate your learning curve and help you succeed in your most challenging software projects. Sessions range from one-hour virtual briefings to half-day and full-day live sessions in cities worldwide.
  - **developerWorks podcasts**
    Listen to interesting and offbeat interviews and discussions with software innovators.
  - **developerWorks on Twitter**
    Check out recent Twitter messages and URLs.
  - **IBM Education Assistant**
    A collection of multimedia educational modules that will help you better understand IBM software products and use them more effectively to meet your business requirements.