Mobile Application Services in IBM WebSphere Application Server Web 2.0 and Mobile Toolkit Version 1.1.0

Contents

Overview of the Mobile Application Services

Mobile Application Services provides a collection of REST-based building blocks that are intended to simplify the creation of mobile-focused applications.

This `85528_WAS_web2mobile-mobile-application-services-help.pdf` file derives from the 8.5.5 `com.ibm.websphere.web2mobile.mobile.application.services.help` plug-in that was used in WebSphere Application Server 8.5.5.28 and previous versions. The `com.ibm.websphere.web2mobile.mobile.application.services.help` plug-in is available in the `855_WAS_all_product_doc.zip` file at `https://public.dhe.ibm.com/software/webserver/appserv/library/v85/`.

Avoid trouble: Mobile Application Services is deprecated. Do not use Mobile Application Services for new applications. For server-side services such as Analytics and Graphics, rearchitect your applications to use Java API for RESTful Web Services (JAX-RS). When you migrate the graphics service, you can use the open source Apache Batik project to convert images. As to support for the Optimizer, WebDAV extension, and Map Conversion application services, there is no recommended migration action. For more information, see statements about the Web 2.0 and Mobile Toolkit deprecated feature in Deprecated features.

The components that are provided include Analytics, Graphics Conversion, and Optimizer services and an Apache Wink WebDAV JAX-RS extension, and samples for adding file upload and directory listing capabilities to your applications. More information about the individual Mobile Application services, samples, and demonstration applications is available on the following pages:

- Analytics service
- Graphics Conversion service
- Optimizer service
- Apache Wink WebDAV JAX-RS extension
- “Getting started with the Map Converter sample application” on page 14

Analytics service

The Analytics service application complements the use of dojox.analytics and related plug-ins from the Dojo Toolkit for JavaScript, providing server-side multiplexing of analytics and logging.

Overview

The Analytics service application was created to complement the use of dojox.analytics and related plug-ins from the Dojo Toolkit for JavaScript. Analytics provides a way to track events (Dojo, mouse clicks) and generate logging from the client-side code. The analytics information is often managed on the client and sent to potentially several different analytics servers. This can be cumbersome and often requires communication with domains outside of the server that provided the application. The Analytics service provides a way to handle the multiplexing of analytics and logging on the server side. The existing dojox.analytics code on the client sends the events to the same server that provided the application, which eliminates any cross domain issues.

On the server, the Analytics service application includes a server-side JAX-RS resource that accepts the requests that are sent from the clients dojox.analytics code. Servlet initialization parameters can be set in the Analytics service application to configure how it handles the analytics events. By default, all events are sent to a single log file, `analytics_default.log`.

Prerequisites

Table 1. Product prerequisites

Product prerequisite	Version
Java Technology Edition	5.0 and later
Java Platform, Enterprise Edition 5 (Java EE) application server and later	WebSphere Application Server Version 8.5
Web browser	Any modern web browser, such as Internet Explorer 7 and later, Mozilla Firefox 3.x and later, Google Chrome, Safari, Opera

Using the Analytics service

The Analytics service is only useful if the client-side JavaScript is using dojox.analytics. To access the tools from dojox.analytics, the client-side JavaScript need only include the proper dojo.require statement. With that dependency pulled in, several options are available to generate events.

Enable the ability to generate events:

```
dojo.require("dojox.analytics.plugins.dojo");
```

Generate a dojox.analytics event with arbitrary text:

```
console.rlog("Any quoted string goes here.");
```

Log something with a high precision timestamp:

```
dojox.analytics.addData("timestamp", [new Date().getTime(), "on-load executed"]);
```

Beyond selecting which events will be tracked by dojox.analytics, configuration to identify the relative or absolute URL of the analytics service is required.

Table 2. djConfig parameters

djConfig parameter	Meaning and possible values	Required?
analyticsUrl:	URL of the analytics service, either fully qualified or relative to the current resource	Yes
sendMethod	'script' or 'xhrPost' (default)	sendMethod: 'script' is required if analytics service is on a different domain than the referring page.
sendInterval	Number of milliseconds to accumulate events before sending to the analytics service	No, defaults to 5000 milliseconds

The default configuration sends all events to a single log file, `analytics_default.log`, located in the directory defined by the server java.io.tmpdir system property from the server JVM. This output log file name and path can be customized via property settings in the `web.xml` file. To do so, you need to specify the full path of the file where you would like events to be logged to, in the param-value for the param-name `com.ibm.ws.mobile.appsvcs.analytics.logger.LocalFileLogger`. You will also need to specify a value of '1.0' to identify the syntax version to be used by the service. This is required, and is simply to allow for versioning in the future.

The following portion of `web.xml` file demonstrates its use:

```
<init-param>
```

```
    <param-name>com.ibm.ws.mobile.appsvcs.analytics.logger.LocalFileLogger</param-name>
    <param-value>1.0,/tmp/events.log</param-value>
</init-param>
```

You can also further customize the format of the log by specifying what other information is recorded as part of each log entry. This can be done via property settings in the `web.xml` file. To do so, simply specify one or more of the predefined keywords in the param-value for the param-name `com.ibm.ws.mobile.appsvcs.analytics.logger.LocalFileLogger.LogFormat`.

The following portion of `web.xml` file demonstrates its use:

```
<init-param>
    <param-name>com.ibm.ws.mobile.appsvcs.analytics.logger.LocalFileLogger.LogFormat</param-
name>
    <param-value>CLIENT_IP,CLIENT_SESSION,HTTP_REFERER</param-value>
</init-param>
```

Table 3. Log format entries

Log format entry	**What is logged**
CLIENT_SESSION	logs HttpServletRequest#getRequestedSessionId(), the request's session ID specified by the client
CLIENT_SESSION_FORCED	logs HttpServletRequest#getSession(true).getId(), forcing the creation of an HTTP session if it doesn't exist, and logging it's session id
CLIENT_IP	logs HttpServletRequest#getRemoteAddr(), the Internet Protocol (IP) address of the client
HTTP_REFERER	logs HttpServletRequest#getHeader(), the request header

REST interface

When a client is using dojox.analytics, the events that it generates result in HTTP requests being sent to an analytics server. The Analytics service processes these requests by exposing a Representation State Transfer (REST) interface.

The Analytics service will handle requests targeted at `http://<server>:<port>/<context-root>/<url-pattern>/analytics/logger`. The context root is defined by the application.xml file if your web archive file (.war) is packaged in an enterprise archive (.ear) file. If the .war is installed by itself, the context root is defined at install time by the user. The uri mapping is defined in the WEB-INF/web.xml in the .war file.

In the provided appsvcs-analytics.ear, the application.xml file specifies a context root of "/appsvcs-analytics" and the web.xml file specifies a URL pattern of "/rest/*".

Table 4. REST operations

Operation Description	**Method**	**URI**	**Parameters**
Report a list of events	GET	/appsvcs-analytics/rest/analytics/logger	There are two required query parameters (id and data) and one that is optional (callback).
Report a list of events	POST	/appsvcs-analytics/rest/analytics/logger	The POST request uses the same parameters, but they are not part of the URI query. Rather, they are in the body of the HTTP request.

Table 5. Query parameters	
Query Parameter	**Description**
id	Unique event identifier
data	JSON formatted array of events
callback	The name of the JavaScript method that should be included in the response.

Table 6. HTTP status codes	
Status Code	**Description**
200	The requested operation succeeded.
400	The request included incorrect values in the parameters or request body.
405	An unsupported URI is in the REST request.
406	The client does not support the required JSON format, per the request headers.
415	The client request includes an unsupported content type.
500	An unexpected error occurred in the server.

Installation

This section describes the procedure for installing the Analytics service on Version 8.5 of the IBM WebSphere Application Server. It is assumed that you are familiar with application installation and administration for the application server.

Before you begin:

Locate the Analytics service enterprise archive (EAR) file that is provided with your product installation. You can find the EAR file in your installation tree where you installed the IBM WebSphere Application Server Web 2.0 and Mobile Toolkit.

Table 7. Installation locations	
Platform	**Location**
Linux and UNIX:	/opt/WebSphere/Web20Mobile/installableApps/application_services/analytics/`appsvcs-analytics.ear`
Windows:	c:\WebSphere\Web20Mobile\installableApps\application_services\analytics\`appsvcs-analytics.ear`

Installation steps:

1. Log into the administrative console for the application server.
2. Navigate to **Applications** > **New Application**. (Note: In WebSphere Application Server Version 6.1, select **Install New Application**)
3. Select **New Enterprise Application**. (Note: In WebSphere Application Server Version 6.1, skip this step)
4. Browse your file system, and select the `appsvcs-analytics.ear` file that you located earlier. Click **Next**.
5. Click **Next** to prepare for the application installation. (Note: In WebSphere Application Server Version 6.1, skip this step)
6. Click **Next** to accept the default installation options.
7. Click **Next** to accept the default options for map modules to servers.

8. Click **Next** to accept the default options for Metadata for modules. (Note: In WebSphere Application Server Versions 6.1 and 7, skip this step)
9. Click **Next** to accept the default options for map virtual hosts for web modules.
10. Review the summary of the installation options.
11. Click **Finish**.
12. Click **Save to the master configuration**.
13. Navigate to **Applications** > **Application Types** > **WebSphere Enterprise Applications**. (Note: In WebSphere Application Server Version 6.1, Navigate to **Applications** > **Enterprise Applications**)
14. Select the **IBM WebSphere Application Server - Analytics Service**, and click **Start**.

Access the installed application demo client:

Install showcase sample for the democlient and point your web browser to your application server installation: http://<application server hostname>:<port>/appsvcs-analytics/

The application server host name and port number are specific to your application server installation. An application server default installation web container port is 9080. If you are running your web browser on the same workstation as your application server installation and have taken all the default values, then use the following URL: http://localhost:9080/appsvcs-analytics/.

Graphics conversion service

The Graphics conversion service provides a server-side JAX-RS resource that supports scaling and conversions to and from many graphics formats.

Overview

The Graphics conversion service provides a server-side JAX-RS resource that supports scaling and conversions to and from many graphics formats. See the conversion chart for details.

Table 8. Graphics format conversion support

From/To	GIF	JPEG	PDF	PNG	TIFF	GFX
GIF	Yes	Yes[1]	No	Yes[1]	No	No
JPEG	Yes[1]	Yes	No	Yes	No	No
PDF	---	---	---	---	---	---
PNG	Yes[1]	Yes	No	Yes	No	No
SVG	No	Yes	Yes	Yes	Yes	Yes[2]
TIFF	---	---	---	---	---	---

[1] Supported only with Java SDK Version 6.0 or higher.

[2] There are limitations in the scope of SVG to GFX function. See the Limitations section.

A sample client application, provided via the Mobile Showcase sample, illustrates a simple Ajax invocation of the Graphics Conversion service. The Mobile Showcase sample is available in the WebSphere Application Server 8.5 samples infocenter.

Note:

- The Graphics conversion sample application is only to illustrate one possible use of the Graphics conversion service, and to illustrate how one would use AJAX invocations to make the REST request.
- Because of limitations existing in certain browsers with respect to rendering SVG content within img and iframe html tags, workarounds are sometimes used which involve rendering non-SVG image when SVG image is selected. However, this is simply for selection display purposes and is so noted within the html document. However, when the actual AJAX request is sent to the service the actual SVG image is always

sent in for conversion. In the results view, you will always see the actual converted image, which can be saved if necessary.

Prerequisites

Table 9. Product prerequisites

Product prerequisite	**Version**
Java Technology Edition	5.0 and later
Java Platform, Enterprise Edition 5 application server and later	WebSphere Application Server Version 8.5
Web browser	Any modern web browser, such as: Internet Explorer 7 and later, Mozilla Firefox 3.x and later, Google Chrome, Safari, Opera

Limitations

- There are processing limitations with non compliant SVG images that do not define a viewBox or height and size. When no viewBox is present, and the image does not specify hardcoded width or height, the Graphics Conversion service does its best to convert the image to requested height and width while at the same time maintaining the aspect ratio of the given SVG document. In some circumstances, users may not get the image that is expected. Therefore, provide compliant SVG images to ensure that the most accurate conversion is performed.
- The SVG to GFX transformations are limited.
- Support for GIF, PDF, and TIFF conversions are limited.

Security considerations

- The Graphics Conversion service writes temporary files to the server file system and returns URLs to these converted images. Application administrators must ensure that proper authorization is provided to users to avoid any security exposure. Care should be taken to make sure that the temporary files do not consume excessive disk space because there is currently no cache control or clean up of these files.
- The service receives a URL reference to the source graphic to be used in the conversion. Administrative care should be taken to assure that this URL is not used as part of a denial of service attack on that remote server or on the local application server.
- The service must load the entire source graphic to perform the conversion. Administrative care should be taken to ensure that there are sufficient resources available to process the selected graphic file. Files of excessive size should be rejected.

Using the Graphics Conversion service

1. To use the Graphics Conversion service, deploy the service .ear file to an application server. Read about Installing the Graphics Conversion service.
2. After the service is deployed, you can use the client demo provided. The demo is available at:

   ```
   http://<server>:<port>/appsvcs-graphics/
   ```

3. Try different values in the various input fields to see the converted images.
4. To directly generate dynamic image conversions in your browser, you can also enter a URL with the parameters that you specify; for example:

   ```
   http://<server>:<port>/appsvcs-graphics/rest/graphics/convert/binaryResponse?
   sourceUrl=<graphic_file_url>&desiredFormat=<new_format>&maxWidth=<optional_width>&maxHeight=<optional_heig
   ```

Note: Working files and converted files returned via URL-reference are stored in a temporary directory location which is specified by the resultsTmpDir configuration parameter that is declared in an initialization parameter of the `web.xml` file. The following portion of `web.xml` file demonstrates its use:

```
<init-param>




    <param-name>com.ibm.ws.mobile.appsvcs.graphics.resultsTmpDir</param-name>
    <param-value>java.io.tmpdir</param-value>
</init-param>
```

Installation

This section describes the procedure for installing the Graphics Conversion service into Version 8.5 of the IBM WebSphere Application Server. It is assumed that you are familiar with application installation and administration for the application server.

Before you begin:

Locate the Graphics Conversion service enterprise archive (EAR) file that is provided with your product installation. You can find the EAR file in your installation tree where you installed the IBM WebSphere Application Server Web 2.0 and Mobile Toolkit.

Table 10. Installation locations

Platform	**Location**
Linux and UNIX:	`/opt/WebSphere/Web20Mobile/installableApps/application_services/graphics/appsvcs-graphics.ear`
Windows:	`c:\WebSphere\Web20Mobile\installableApps\application_services\graphics\appsvcs-graphics.ear`

Installation steps:

1. Log into the administrative console for the application server.
2. Navigate to **Applications** > **New Application**. (Note: In WebSphere Application Server Version 6.1, select **Install New Application**)
3. Select **New Enterprise Application**. (Note: In WebSphere Application Server Version 6.1, skip this step)
4. Browse your file system, and select the `appsvcs-graphics.ear` file that you located earlier. Click **Next**.
5. Click **Next** to prepare for the application installation. (Note: In WebSphere Application Server Version 6.1, skip this step)
6. Click **Next** to accept the default installation options.
7. Click **Next** to accept the default options for map modules to servers.
8. Click **Next** to accept the default options for Metadata for modules. (Note: In WebSphere Application Server Versions 6.1 and 7, skip this step)
9. Click **Next** to accept the default options for map virtual hosts for web modules.
10. Review the summary of the installation options.
11. Click **Finish**.
12. Click **Save to the master configuration**.
13. Navigate to **Applications** > **Application Types** > **WebSphere Enterprise Applications**. (Note: In WebSphere Application Server Version 6.1, Navigate to **Applications** > **Enterprise Applications**)
14. Select the **IBM WebSphere Application Server - Graphics Conversion service**, and click **Start**.

Optimizer service

The Optimizer service is a JAX-RS based service that delivers Dojo Toolkit for JavaScript resources to clients in an optimized form, including content for specific user-agents, compression, and customizable HTTP caching and expiration.

Overview

The Optimizer service is a JAX-RS based service that delivers Dojo Toolkit for JavaScript resources to clients in an optimized form, including content for specific user-agents, compression, and customizable HTTP caching and expiration.

The Optimizer service sets HTTP caching and expiration data based on a ratio of the age of the on-disk resources, and saves compressed representations of resources to serve subsequent compression-aware clients.

The Optimizer service also automatically selects pre-built distributions of Dojo tailored to specific user-agents when available.

Prerequisites

Table 11. Product prerequisites

Product prerequisite	**Version**
Java Technology Edition	5.0 and later
Java Platform, Enterprise Edition 5 (Java EE) application server and later	WebSphere Application Server Version 8.5
Web browser	Any modern web browser, such as: Internet Explorer 7 and later, Mozilla Firefox 3.x and later, Google Chrome, Safari, Opera

Using the Optimizer service

The Optimizer service has the Dojo Toolkit for JavaScript from the Web 2.0 and Mobile Toolkit already packaged within the .war file packaged in the `appsvcs-optimizer.ear` file. The application is immediately installable and deployable. However, if you want to maintain your own Dojo Toolkit for JavaScript elsewhere on the file system of the server on which this application is installed, you can perform the following steps, which are optional:

1. Extract the .war file from the `appsvcs-optimizer.ear` file.
2. Extract the `WEB-INF/web.xml` file from the .war file.
3. Edit the `WEB-INF/web.xml` file. Set the param-value for `com.ibm.ws.mobile.appsvcs.optimizer.srcPath` to the full path of the file system directory in which you have the folder named "dojo", which contains the Dojo you intend to make available via the Optimizer service.
4. Package the `WEB-INF/web.xml` file back into the .war file.
5. Package the .war file back into the `appsvcs-optimizer.ear` file.

Dojo Toolkit for JavaScript has a builder that can produce optimized builds. These too can be hosted under the "dojo" directory to which you have configured the Optimizer service to use. For example, you might want to host an older version of Dojo. Therefore, you might also create the directory, "dojo/1.5/".

There is a single special case that Optimizer service currently supports. The Dojo builder supports a webkitMobile flag that produces a Dojo build that is optimized for WebKit-based Web browsers. The Optimizer service will use the User-Agent header of any incoming request to detect if it is a WebKit client, and look for the requested file in a directory named `dojo_webkit` that is a peer to the `dojo` directory.

Existence of the `dojo_webkit` directory is entirely optional, but if it exists and contains a Dojo build that used the webkitMobile=true flag, you might achieve slightly better performance on WebKit based clients.

To use the Optimizer service, perform the following steps:

1. Install the Optimizer service enterprise archive (.ear) file.
2. Update the references to Dojo resources (CSS and JavaScript) in your web application to point to the Dojo files beneath the Optimizer service context root followed by "rest/optimizer/" (for example, /appsvcs-optimizer/rest/optimizer/dojo/dojo.js).

Configuring the Optimizer service

The Optimizer service reads a number of optional ServletConfig parameters that can be configured using the `META-INF/web.xml` file.

Table 12. Configuration parameters

Parameter	**Description**
`com.ibm.ws.mobile.appsvcs.optimizer.srcPath`	Identifies the absolute file system path of the dojo build. Default: `dojo/` (located under the WAR file)
`com.ibm.ws.mobile.appsvcs.optimizer.cacheDeltaFactor`	The Optimizer service sets the HTTP cache expiration time of resources based on a fraction of how old the on-disk representation is. This directive specifies the floating point number that the age will be divided by to determine how long clients will be permitted to cache a file. Values: Floating point literal to divide the age Default: 1000f, .01% of the age of the file

Table 12. Configuration parameters (continued)

Parameter	Description
`com.ibm.ws.mobile.appsvcs.optimizer.cacheDeltaMin`	When `Expires` or `Cache-Control` headers are issued by the service, this setting dictates the minimum number of seconds in the future that the resources remain cacheable. Recently updated files will be cached by clients for at least the amount of time specified in this directive. Attention: When a HTTP client caches a resource, there is no way to invalidate that at a subsequent date; therefore take care in increasing this number. Values: Number of seconds Default: 5
`com.ibm.ws.mobile.appsvcs.optimizer.cacheDeltaMax`	When `Expires` or `Cache-Control` headers are issued by the service, this setting dictates the maximum number of seconds in the future that the resources remain cacheable. Files that have not been updated for long periods of time will be cacheable by clients for no longer than the number of seconds specified in this directive. Values: Number of seconds Default: 900
`com.ibm.ws.mobile.appsvcs.optimizer.sendExpires`	Controls whether the Optimizer service sends the HTTP `Expires` header used by private and shared caches. Values: true or false Default: true

Table 12. Configuration parameters (continued)

Parameter	Description
`com.ibm.ws.mobile.appsvcs.optimizer.sendETAG`	Controls whether the Optimizer service sends the HTTP `ETag` header used by private and shared caches. Values: true or false Default: true
`com.ibm.ws.mobile.appsvcs.optimizer.sendCCMaxAge`	Controls whether the Optimizer service sends the HTTP `Cache-Control` header with the `max-age` parameter, used by private caches. Values: true or false Default: true
`com.ibm.ws.mobile.appsvcs.optimizer.noVary`	Controls whether the Optimizer service sends the HTTP `Vary` header to indicate that the response took into account headers such as `User-Agent` and `Accept-Encoding`. Values: true or false Default: false

Installation

This section describes the procedure for installing the Optimizer service on Version 8.5 of the IBM WebSphere Application Server. It is assumed that you are familiar with application installation and administration for the application server.

Before you begin:

Locate the Optimizer service enterprise archive (EAR) file that is provided with your product installation. You can find the EAR file in your installation tree where you installed the IBM WebSphere Application Server Web 2.0 and Mobile Toolkit.

Table 13. Installation locations

Platform	Location
Linux and UNIX:	/opt/WebSphere/Web20Mobile/installableApps/application_services/optimizer/`appsvcs-optimizer.ear`
Windows:	c:\WebSphere\Web20Mobile\installableApps\application_services\optimizer\`appsvcs-optimizer.ear`

Installation steps:

1. Log into the administrative console for the application server.

2. Navigate to **Applications** > **New Application**. (Note: In WebSphere Application Server Version 6.1, select **Install New Application**)
3. Select **New Enterprise Application**. (Note: In WebSphere Application Server Version 6.1, skip this step)
4. Browse your file system, and select the `appsvcs-optimizer.ear` file that you located earlier. Click **Next**.
5. Click **Next** to prepare for the application installation. (Note: In WebSphere Application Server Version 6.1, skip this step)
6. Click **Next** to accept the default installation options.
7. Click **Next** to accept the default options for map modules to servers.
8. Click **Next** to accept the default options for Metadata for modules. (Note: In WebSphere Application Server Versions 6.1 and 7, skip this step)
9. Click **Next** to accept the default options for map virtual hosts for web modules.
10. Review the summary of the installation options.
11. Click **Finish**.
12. Click **Save to the master configuration**.
13. Navigate to **Applications** > **Application Types** > **WebSphere Enterprise Applications**. (Note: In WebSphere Application Server Version 6.1, Navigate to **Applications** > **Enterprise Applications**)
14. Select the **IBM WebSphere Application Server - Optimizer service**, and click **Start**.

Access the installed application demo client:

Point your web browser to your application server installation:

```
http://<application server hostname>:<port>/appsvcs-optimizer/
```

The application server host name and port number are specific to your application server installation. An application server default installation web container port is 9080. If you are running your web browser on the same workstation as your application server installation and have taken all the default values, then use the following URL: http://localhost:9080/appsvcs-optimizer/.

Apache Wink WebDAV JAX-RS extension

The Apache Wink WebDAV JAX-RS extension library helps JAX-RS application developers build WebDAV-compliant services by providing a JAXB data model, helper classes, and JAX-RS based extensions.

Overview

Web-based Distributed Authoring and Versioning (WebDAV) is a defined set of HTTP methods and responses to allow users to read and write documents on a web server. Most websites today use basic HTTP to allow users to easily read data by using HTTP GET methods and to create and modify data by using HTTP POST and PUT methods. WebDAV enhances the ability to manage, edit, and write documents by providing features to set and read document properties, lock resources, and support for collections. WebDAV defines a set of HTTP methods and request and response entity formats to form the WebDAV protocol.

The Apache Wink WebDAV JAX-RS extension library helps JAX-RS application developers build WebDAV-compliant services. It provides a JAXB data model of the WebDAV XML to help read requests and write responses, helper classes to build responses, and JAX-RS based extensions to more easily build WebDAV services.

The JAXB model is provided in the `org.apache.wink.webdav.model` package. Helper classes include `org.apache.wink.webdav.server.WebDAVResponseBuilder`, which builds appropriate WebDAV responses with various properties. In the `org.apache.wink.webdav` package, JAX-RS compatible @HttpMethod annotations are available for the WebDAV HTTP methods and constants for common WebDAV HTTP headers and properties. See the API documentation for more details.

Important: The JAX-RS WebDAV extension is included in the WebSphere Application Server runtime. The JAX-RS WebDAV extension is not installed by the Web 2.0 and Mobile Toolkit.

Migration

If you are migrating from the WebSphere Application Server Version 7 or Version 8 Feature Pack for Web 2.0 and Mobile to WebSphere Application Server V8.5, there are a few things to consider:

- The WebDAV JAX-RS extension is now part of the WebSphere Application Server runtime. There are no WebDAV Java Archive (JAR) files included with the Web 2.0 and Mobile Toolkit or WebSphere Application Server.
- When moving a WebDAV enabled application to WebSphere Application Server V8.5, remove the `webdav` jar from your web application.

Prerequisites

Read the IBM JAX-RS documentation for more information about configuring JAX-RS. You can check the JAX-RS documentation by clicking the appropriate link:

../../was/ae/cwbs_jaxrs_overview.dita JAX-RS Documentation.

Using the Apache Wink WebDAV JAX-RS extension library

You can use the WebDAV extension library to write JAX-RS methods and resources that can communicate using the WebDAV protocol.

Here is a sample resource method:

```
@WebDAVMethod.PROPFIND
@Produces(MediaType.APPLICATION_XML)
public Response findProperties() throws IOException {
     SyndFeed feed = /* create an Apache Wink SyndFeed */;
     return WebDAVResponseBuilder.propfind(feed);
}
```

The previous method responds to any HTTP PROPFIND method requests, and using the WebDAVresponseBuilder, the resource method returns a response with the feed data.

A JAXB model is provided in the org.apache.wink.webdav.model Java package to help write responses and read requests. For instance, if you want to parse a WebDAV lock request, you could use the following to read the XML:

```
String bodyContent = /* the XML request */
Unmarshaller unmarshaller = WebDAVModelHelper.createUnmarshaller();
Lockinfo lockinfo = WebDAVModelHelper.unmarshal(unmarshaller,
                                                new StringReader(bodyContent),
                                                Lockinfo.class,
                                                "lockinfo");
```

In addition, you can use a few resource classes to help you get started with creating WebDAV resources.

WebDAVResource provides an @OPTIONS method that adds the WebDAV appropriate headers:

```
@Path("/myresource/")
public class MyResource extends org.apache.wink.webdav.server.WebDAVResource {
    /* get an @OPTIONS annotated JAX-RS resource method implementation from WebDAVResource */

    /* implement your own WebDAV methods */
}
```

WebDAVLockableResource provides a few stub methods for performing WebDAV locking and unlocking. They do not perform a real lock but provide a fake lock to satisfy compatibility with certain platforms:

```
@Path("/myresource2/")
public class MyResource2 extends org.apache.wink.webdav.server.WebDAVLockableResource {
    /* get @LOCK and @UNLOCK stub methods for compatibility */
```

```
    /* implement your own WebDAV methods */
}
```

Getting started with the Map Converter sample application

This document is the starting point for learning about the Map Converter sample application.

The following topics are covered:

- Overview - Introduces the Map Conversion service and the sample application.
- Prerequisites - Indicates the products and versions required to use the service.
- Limitations of the Map Converter application - Mentions the limitations of the sample application.
- Security considerations - Lists the security considerations that apply when using the Map Conversion service.
- Using the Map Converter - Describes how to use the Map Converter.
- Installing the Map Conversion service - Describes version-specific ways of installing the Map Conversion service with the sample application.

Overview

The Map Converter sample application demonstrates the use of the Map Conversion service.

The Map Converter sample application exists to show the use of the Map Conversion service. The Map Conversion service provides a server-side JAX-RS resource that supports conversion from cartographic ESRI shapefiles to a JSON representation of the map, suitable for display by a Dojo `dojox.geo.charting` widget.

A sample client application, provided by the Mobile Showcase sample on WebSphere Application Server, uses the Map Conversion service. Users of this application can upload a set of ESRI cartographic files to the server, choose parameters such as projection and simplification ratio, and download the resulting JSON map file. The JSON map file can be used as a feed for a `dojox.geo.charting` application.

Related concepts

Next topic
This topic describes the products and versions required to use the Map Conversion service.

Prerequisites

This topic describes the products and versions required to use the Map Conversion service.

Table 14. Prerequisites

Product prerequisite	**Version**
Java Platform, Enterprise Edition 5 (Java EE) application server and later	WebSphere Application Server Version 8.5
Web browser	Any modern web browser, such as: Mozilla Firefox 3.x and later, Google Chrome, Safari, Opera

Related concepts

Next topic
The Map Converter application has limitations that restrict its use to development and educational purposes.

Limitations of the Map Converter application

The Map Converter application has limitations that restrict its use to development and educational purposes.

The Map Converter application is not intended for deployment to production servers. It is for development and educational purposes only.

Related concepts

Next topic

Several security considerations apply when using the Map Conversion service.

Security considerations

Several security considerations apply when using the Map Conversion service.

The following security considerations exist:

- The Map Converter application uploads files to the server file system and returns URLs that refer to these uploaded files. Application administrators must ensure that proper authorization is provided to users to avoid any security exposure. Care should be taken to make sure that the uploaded files do not consume excessive disk space, because currently cache control is **not** applied to these files.
- The service receives a URL reference to the source map files to be used in the conversion. Administrative measures should be taken to make sure that this URL is not used as part of a denial of service attack on that remote server or on the local application server.
- The application does not implement any authentication process. Therefore, all the uploaded files are accessible to any user connected to the service. The application can delete uploaded files, even those that have been uploaded by other users connected to the server.

Related tasks

Next topic

The Map Converter converts ESRI shapefiles to a JSON format suitable to be displayed in a Dojo geocharting component.

Using the Map Converter

The Map Converter converts ESRI shapefiles to a JSON format suitable to be displayed in a Dojo geocharting component.

About this task

The Map Converter is a Dojo application connected to a REST service running on a server. Computation is done on the server. The converted files can be queried by the client.

The server implements some services, such as uploading files to the server, reprojecting map files, simplifying map files, and converting map files into JSON format.

Procedure

Provide input parameters

1. Upload shape files to the server

   This action must be performed first to be able to use the Map Converter.

   a) In the **Conversion Settings** panel, click the **Upload** button.

      Or

      In the **Uploaded Files** panel, click the **Upload** button.

   b) Select the files you want to upload.

      For each map, the Map Converter requires at least two files, the `.shp` file and a `.dbf` file. The `.shp` file contains the shapes of the map features. The `.dbf` file contains metadata associated with each map feature. To select multiple files, hold the control key while you select the files.

Multiple selection is not available for Internet Explorer browsers. If you are using Internet Explorer browser, you must upload each file one by one.

c) Upload the files.

Click **Open** to upload the selected files.

You can see the list of uploaded files on the server in the **Uploaded Files** panel. This list is not a per-session or authenticated file list. It shows all the files uploaded on the server, including the files uploaded by other clients.

This panel provides an option to delete files from the server. If you want to delete files on the server, delete only files that you have uploaded. To delete a file, select it and click the delete button.

2. Select the file to convert

In the **Conversion Settings** panel, from **Shapefile**, select the file that you want to convert.

3. Associate metadata with map features

In **Feature name**, select metadata that is appropriate to your application.

You select the metadata to be associated with each map feature to identify the feature and to bind it with the data store in the `dojox.geo.charting` widget. Values corresponding to the metadata are displayed in the Map Preview pane as tooltips. For example, selecting NAME causes the ISO country identifiers to be displayed as tooltips.

4. Select input projection

Select the appropriate input projection.

Usually the input projection is the projection labeled **Longitude/Latitude**, which is a geographic position. If the Shapefile has been encoded with a different projection, select that projection from the list.

Provide output parameters

5. Select and test the output projection

a) Select the output projection

Select the output projection that seems appropriate for your map.

This parameter specifies the projection in which the converted map is projected. If you select the same projection as the input projection, the map will not be projected. Not all the listed projections are valid for every map. Unexpected results can occur when unsuitable projections are selected.

b) Visualize the result

Click **Convert**.

The converted map is displayed in the pane on the right.

6. Select type of coordinates

Select integer or floating-point coordinates for the converted JSON map.

Floating-point coordinates are selected by default. Integer coordinates must be explicitly selected. Selecting integer coordinates usually results in a smaller JSON file. In this case, you might have to choose a **Scale Map to** value for the map to increase precision. The default value usually works well.

7. Select Minimum detail size

Enter **Minimum detail size** to obtain a lighter, but less detailed map.

The minimum detail size is expressed in meters. It indicates that no details in the converted map will be larger than this distance. This parameter is used to obtain a smaller JSON file.

8. Remove unwanted artifacts

Select **Wrap dateline**.

Under certain circumstances, projecting a map can produce some unwanted artifacts, especially when projecting a world map. This option can remove such glitches, but requires more time to project the map.

Download converted file for use in an application

9. Download the file

When you are satisfied with the results of the conversion, click **Download** to download the converted JSON data.

10. Integrate the data in an application

    Integrate the data generated by the Map Converter as map description data in a `dojox.geo.charting` widget.

    The following example assumes that you downloaded the JSON data as a file named `map.json` and shows how to use it.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">;
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Use Map Converter Data</title>esources/UploaderFileList.css">
<script type="text/javascript">
 var djConfig = {
  parseOnLoad : true
 }
</script>
<script type="text/javascript" src="../../../dojo_current/dojo/dojo.js"></script>
<script type="text/javascript">
 require([ "dojox/geo/charting/widget/Map" ]);
</script>

</head>

<body class="claro">
 <div data-dojo-type="dojox.geo.charting.widget.Map"
  shapeData="map.json" />
</body>
</html>
```

Related concepts

Next topic
Describes version-specific ways of installing the Map Conversion service with the sample application.

Installing the Map Conversion service

Describes version-specific ways of installing the Map Conversion service with the sample application.

The following topic describes how to install the Map Conversion service:

- In WebSphere Application Server - Describes the procedure for installing the Map Conversion service in Version 8.5 of the IBM WebSphere Application Server.

Related tasks

Previous topic
The Map Converter converts ESRI shapefiles to a JSON format suitable to be displayed in a Dojo geocharting component.

Next topic
Describes the procedure for installing the Map Conversion service in Version 8.5 of the IBM WebSphere Application Server.

In WebSphere Application Server

Describes the procedure for installing the Map Conversion service in Version 8.5 of the IBM WebSphere Application Server.

Before you begin

You should be familiar with application installation and administration for the application server (Version 8.5 of the IBM WebSphere Application Server).

About this task

Locate the Map Conversion web application archive (WAR) file that is provided with your product installation. You can find the WAR file in your installation tree where you installed the IBM WebSphere Application Server Web 2.0 and Mobile Toolkit.

For example, if you installed the toolkit in the following location:

Table 15. Platform Install Location

Platform	**Location**
Linux and UNIX:	`/opt/WebSphere/Web20Mobile`
Windows:	`c:\WebSphere\Web20Mobile`

Then, you can find the WAR file at:

Table 16. WAR Install Location

Platform	**Location**
Linux and UNIX:	`/opt/WebSphere/Web20Mobile/installableApps/application_services/mapconverter/dojo-map-converter-server.war`
Windows:	`c:\WebSphere\Web20Mobile\installableApps\application_services\mapconverter\dojo-map-converer-server.war`

Procedure

Installing the Map Conversion service using the administrative console

1. Log into the administrative console of the application server.
2. Navigate to **Applications** > **New Application** or in WebSphere Application Server Version 8.5, select **New Application**.
3. Select **New Enterprise Application**.
4. Browse your file system and select the `dojo-map-converter-server.war` file that you located earlier; click **Next**.
5. Click **Next** to prepare for the application installation.
6. Click **Next** to accept the default installation options.
7. Click **Next** to accept the default options for map modules to servers.
8. Click **Next** to accept the default options for metadata for modules.
9. Click **Next** to accept the default options for map virtual hosts for web modules.
10. Review the summary of the installation options.
11. Click **Finish**.
12. Save to the master configuration.
13. Navigate to **Applications** > **Application Types** > **WebSphere Enterprise Applications**
14. Select IBM WebSphere Application Server — Map Conversion service and click **Start**.

Related concepts

Previous topic
Describes version-specific ways of installing the Map Conversion service with the sample application.

Index