Volantis Mobility Server

Media Access Proxy™

ver. 5.1

Installation & Administration Guide

Volantis Mobility Server

Media Access Proxy™

Version 5.1

Installation and Administration Guide

Media Access Proxy: Installation and Administration Guide

Version 5.1
Published date 2009-01-29

Included software

Apache Software Foundation
Free Software Foundation
Jaxen Project
Sun Microsystems Inc

Table of contents

Media Access Proxy

The Media Access Proxy (MAP) is a web application that is used to perform internal transcoding and transformation operations on image files, and on other media types externally using an STI transcoder service. The resources to be processed are referenced in XDIME 2 with the `object` element. MAP also handles XDIME content and ICS URLs in a backward compatible manner.

The MAP application has four principal components.

- Media agent
- Descriptor store
- Operation engine
- ICS servlet

Media agent

When MCS finds an `object` element in the XDIME source, the `src` attribute value is passed to the media agent, together with configuration parameters that will affect the transcoded output. For example, an image configuration would contain a number of parameters that define the result of the operation such as width, height, and image type.

The media agent requests the unique identifier associated with the parameters from the descriptor store, encodes it as a MAP URL and returns the URL to MCS. MCS replaces the original `src` attribute value with the new URL, and renders the page.

Subsequent requests for the same configuration will return the same URL, unless the descriptor store entry has timed out.

Descriptor store

The descriptor store associates unique identifiers with the parameters required by a particular operation.

The store looks for a requested identifier and returns it to the media agent. If there is no match, an entry is added and the new identifier is returned. As entries are inserted or retrieved, timestamps on those entries are updated. A configurable timeout value is used to determine when an entry expires.

Operation engine

When MAP receives a request from a device, it extracts the identifier, and requests the transcoded asset from the operation engine. The operation engine dispatches the request to the registered transcoder plugins for processing.

Transcoder plugins

MAP comes with the following plugins.

Image Processor
: Transcodes and modifies a wide variety of image types. All images are handled internally.

Standard Transcoding Interface (STI)
: Allows MAP to convert simple requests into complex STI requests that can be directed to a specified STI provider

Identity
: Returns information about the received request to the client

When servicing a request, *all* plugins are asked if they wish to handle it. The results returned to the client are those provided by the first plugin that claims to handle the request. Once a plugin accepts a request, no attempt is made to check other plugins. So if the first plugin subsequently fails, the client will receive an error message.

The plugin generates the media resource content, and MAP returns the content to the device.

Note: To process audio and video resources using an STI transcoder, as distinct from using a trancoding service, you will have to develop custom plugins.

ICS servlet

A separate servlet, takes ICS native and transforce URLs and internally invokes the built in image processing plugin.

ICS supports the use of convertible image variants. Each set of convertible image variants resident in a particular location should be assigned an asset group. The actual images do not need to reside on the ICS server. The asset group must indicate the URL of the ICS server, not the variant location itself. The server for the actual image variants may be indicated using appropriate URL parameters or the default image host will be used as specified in the `ics-config.xml` file. If using a non-default image host, the image host name should be provided using either the *v.imgHost* or the *tf.source.host* parameter.

When a convertible image variant is rendered at runtime, the variant URL is created dynamically using the asset group, variant value and the associated transcoding rule specified for the target device in the MCS device repository.

Installing MAP

Media Access Proxy (MAP) has a separate installer. However, there is an option in the MCS installer to set the URL of the MAP web application. After MAP is installed you should review the features of the ICS servlet, and configure the parameter and log settings as required.

Required software

MAP requires a JNDI datasource to be configured in your application server. To enable caching of transcoded assets, you should install the Squid Internet Object Cache server. You can obtain Squid from `http://www.squid-cache.org/`.

Install options

The Image Conversion Service (ICS) can transcode images between various image formats. It can also perform scaling and watermarking operations. Installation of this module will also install a servlet that allows MAP to behave like earlier versions of ICS.

The Example Service is a simple plugin service that just returns the request URL.

The Standard Transcoder Service (STI) plugin allows MAP to convert simple GET requests into complex STI requests that can be directed to a specified STI provider.

Running the installer

The installation wizard steps you through a series of pages. Most of the installer values have defaults. When you have completed a section, click **Next** to move to the next page, or click **Previous** to review or modify any values.

The installer uses the entries to write to the related section of the configuration file. You can edit this file at any time after installation if you need to change the settings.

When the installation is complete you can choose to save an automated installation file at a convenient location. Then you can repeat the installation on another machine with the path to the configuration file as a parameter.

The following steps assume that you have selected all the install options.

1. Start the installer with the command
   `java -jar map_installer-5.1.jar`
2. Choose the *Accept* option to accept the license agreement
3. Select the required install packs
4. On the next page, accept the default server *URL* and *Port* values or enter new settings
5. Optionally add a *Watermark URL*
6. Accept the *In memory IO buffering* and *ICS native mode* settings
7. Enter an *Originator ID* and *STI service URL*
8. Accept the default installation directory or choose an alternate location
9. The installer displays a summary of all installation choices. Use the **Previous** button to go back and change any options. When you are satisfied with all your choices, click **Install** to start the installation, and **Quit** to close the installer.
10. Optionally save your installation settings by clicking **Generate an automatic installation script**, and naming the file. To repeat the installation run the following command.
    `java -jar map_installer-5.1.jar [configuration_file]`
11. Copy the `[install_directory]/webapps/map` directory to your web application server.
12. Follow the instructions in *Configuring MAP*

Testing your installation

Now test that MAP has been properly installed by converting a `.gif` image.

```
http://localhost:8080/map/ics/images/cj24/volantis.gif
```

Configuring MAP

The Media Access Proxy (MAP) is a web application that is used to perform transcoding and transformation operations on media files referenced by XDIME 2. The contained ICS servlet component handles XDIME content and ICS styled URLs in a backward compatible manner.

Configuring MCS

The `media-access-proxy` section contains the location in the `url-prefix` attribute.

The location of the MAP web application is initially defined in the `mcs-config.xml` file, when you install MCS. If the location changes you must edit the `media-access-proxy` element.

```
<media-access-proxy url-prefix="http://host:port/map/map"/>
```

Note: The `map` subdirectory is necessary because MAP consists of a number of servlets in the same web application. The first `map` directory name may vary depending on your installation. The second directory contains the map servlet, and its name is fixed unless modifications are made to the `web.xml` file in the MAP web application.

The ICS servlet is mapped to the `/map/ics/` URL.

STI plugin

The Standard Transcoder Interface (STI) plugin allows transcoding requests to be forwarded to sites providing STI compliant transcoding services. The plugin is configured in the file `com.volantis.map.sti.impl.STIOperation` located in the `WEB-INF/osgi/config` directory. An example follows.

In the words of the STI specification, the value associated with `com.volantis.map.sti.OriginatorID` is

A unique ID which represents the Application Platform that the request originated from. This parameter MAY be used for tracking, and it SHOULD NOT affect the Transcoding process.
The value of `com.volantis.map.sti.ServiceURL` is the URL to the STI compliant transcoding platform.

```
com.volantis.map.sti.OriginatorID=id-from-sti-service
com.volantis.map.sti.ServiceURL=http://localhost:8080/sti
```

Configuring a JNDI datasource

MAP requires a JNDI datasource to be configured in your application server. The JNDI name is 'jdbc/descriptorstore'. The database at this location will automatically be populated with tables when MAP is started for the first time. Both MAP and MCS must have access to the datasource with this name.

The configuration of JNDI bound datasources is application server specific; refer to your web application server documentation for more information.

The following example shows the settings for Tomcat 5.5 and a Derby database configured on the localhost at port 3333. The resource name is the JNDI name of the datasource. This configuration assumes that MCS is installed under the `/volantis` servlet prefix and MAP is installed under the `/map` servlet prefix. For this to work the `.jar` file containing the `org.apache.derby.jdbc.ClientDriver` class must be visible to both servlet contexts. This can be achieved by placing a single copy in the `common/lib` directory of Tomcat or by placing one copy each into the `WEB-INF/lib` directory of each web application.

Add the following to the `conf/server.xml` inside the appropriate `Host` element.

```
<Context path="/map" debug="99">
  <Resource name="jdbc/descriptorstore" auth="Container"
    type="javax.sql.DataSource" username="sa" password="sa" maxIdle="20"
    driverClassName="org.apache.derby.jdbc.ClientDriver"
    url="jdbc:derby://127.0.0.1:3333/MAP;create=true" maxActive="30"/>
</Context>
<Context path="/volantis" debug="99">
  <Resource name="jdbc/descriptorstore" auth="Container"
    type="javax.sql.DataSource" username="sa" password="sa" maxIdle="20"
    driverClassName="org.apache.derby.jdbc.ClientDriver"
    url="jdbc:derby://127.0.0.1:3333/MAP;create=true" maxActive="30"/>
</Context>
```

Caution: Owing to an issue with PostgreSQL, MAP will not automatically create the necessary tables for the descriptor store. First, extract the `synergetics.descriptor-store-impl-5.0.jar` to get the `package.jdo` file. You can then resolve the problem by using the following script.

```
[path_to_mcs]/webapps/mcs/WEB-INF/lib$ java -cp
log4j-1.2.8.jar:jdo2-api-2.0.jar:jpox-1.1.7.jar
:[path_to_postgres_driver]/postgresql-8.2-506.jdbc3.jar
:synergetics.descriptor-store-impl-5.0.jar
:synergetics.descriptor-store-api-5.0.jar
:synergetics.metadata-impl-5.0.jar
:synergetics.metadata-api-5.0.jar
-Djavax.jdo.option.ConnectionDriverName=org.postgresql.Driver
-Djavax.jdo.option.ConnectionURL="jdbc:postgresql://localhost:5432/MAP"
-Djavax.jdo.option.ConnectionUserName=MAP
-Djavax.jdo.option.ConnectionPassword=MAP
-Dlog4j.configuration=file:../mcs-log4j.xml org.jpox.SchemaTool
-create [path_to_mcs]/webapps/mcs/WEB-INF/[path_to_jdo]/impl/package.jdo
```

If you are using MCS with MAP on WebLogic or WebSphere, then you must add the following to `[path]/MAP/webapps/map/WEB-INF/web.xml` and `[path]/MCS/webapps/mcs/WEB-INF/web.xml` at the end of the `web-app` section. This is to properly configure a JNDI datasource.

```
<resource-ref>
  <description>DB Connection</description>
  <res-ref-name>jdbc/descriptorstore</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
</resource-ref>
```

Editing the ICS configuration file

You set the ICS configuration options in the `ics-config.xml` file. You can specify defaults for the location of images, and how they should be transcoded. See *ICS configuration parameters* for details of the parameters.

An `init-param` element in the `web-xml` file gives the path to the ICS configuration file. The `param-name` attribute names the file and the `param-value` attribute gives the path information.

```
<servlet>
  <servlet-name>ICS</servlet-name>
  <servlet-class>com.volantis.transcoder.Server</servlet-class>
  <init-param>
    <param-name>config.file</param-name>
    <param-value>/WEB-INF/ics-config.xml</param-value>
  </init-param>
</servlet>
```

The example shows the default entries in the `ics-config.xml` file..

```
<transcodingServer>
  <image jpegFormat='baseline' gifSupport='true'>
    <environment serverMode='native'
      maxRedirects="5"
      useInMemoryChache='true' />
    <scaling scaleMode='bilinear' scaleLarger='true'/>
    <dithering>
      <dither bitDepth='1' mode='floyd-steinberg'/>
      <dither bitDepth='2' mode='ordered-dither'/>
      <dither bitDepth='4' mode='stucki'/>
      <dither bitDepth='8' mode='jarvis'/>
      <dither bitDepth='16' mode='floyd-steinberg'/>
    </dithering>
    <compression qualityMin='50' paletteMin='4'/>
    <watermarking watermarkURL=''/>
  </image>
</transcodingServer>
```

Note: If you set the optional `imgHost` and `imgPort` attributes, ICS assumes that there is an image server running on the same server as ICS. If you set the `proxyHost` and `proxyPort` attributes, ICS assumes that you have enabled image caching.

Proxying

ICS effectively acts as a proxy between the user agent and the server holding the original image, and passes all headers from the user agent to the server containing the image, with the exception of the accept header.

The accept header is built up of all the media types that ICS supports, which may be affected by configuration options, for example the `gifSupport` attribute.

Caching

ICS supports the caching of rendered images. The caching model obeys HTTP cache control directives.

A large image cache is supported that allows memory and file-based caching, with migration between levels based on usage. Both source and rendered images may be held in the same cache, or separate ones may be used.

Watermarking

You can configure ICS to overlay a transparent watermark on images by setting the `watermarkURL` attribute on the `watermark` element. The setting defines a single system-wide URL that contains the watermark. The attribute is empty by default. If the attribute is not set, the watermarking feature is turned off.

Note: In some circumstances, watermarking will not be applied. If the device can display the image without transcoding, no processing takes place and the image is sent directly to the target device.

Disabling host and port parameters

If the ICS servlet must for some reason be configured with the `imgHost` and `imgPort` attributes set to an internal server address, you can disable their use in requests to avoid a potential security breach. To do so, you should set the `ignoreHostParameterInRequestUrl` attribute on the `environment` element to 'true'. If the attribute is not set or has a value of 'false', ICS will accept the parameters.

Logging and error handling

Another `context-param` element in the `web-xml` file gives the path to the log configuration file. The `param-name` attribute names the file and the `param-value` attribute gives the path information.

```
<context-param>
  <param-name>map.log4j.config.file</param-name>
  <param-value>/WEB-INF/map-log4j.xml</param-value>
</context-param>
```

The `ics-log4j.xml` file has a path for the log file added to it. You may want to change this value to a more suitable location.

ICS logs any remote and local image server failures that can arise from a request. If a remote server returns a standard HTTP status code, ICS passes it through to the client. If the remote server can not be contacted, the 404 status is passed back. If the local server encounters errors when processing the URL request, or performing the transcoding operation, an HTTP status of 500 is returned.

If you are using JBoss, then make sure that your `jboss-service.xml` file contains the following settings. That is to ensure the environment is compatible with the Java Servlet specification.

```
<attribute name="Java2ClassLoadingCompliance">false</attribute>
<attribute name="UseJBossWebLoader">false</attribute>
```

Accelerated performance

The MAP installer includes the Java Advanced Imaging (JAI) core libraries `jai_codec.jar` and `jai_core.jar`, located under `webapps/map/WEB-INF/osgi/bundles`. These libraries provide all the basic image transcoding functions, but are less performant than native language acceleration libraries that are available for some platforms. If a native implementation is present, JAI will choose it for any available function.

To improve transcoding you can download the native language libraries for your platform from *Java Advance Imaging API*, provided that it can support these libraries.

ICS features

The ICS servlet can convert images to a different format and pixel depth, reduce the color depth for a smaller color palette, and scale images. In addition to these features, you can specify the desired size of the target image which will have a potential bearing on all these factors, as well as general compression if applicable.

Color depth conversion

ICS adapts the color representation used in the output image to the output file type, and the device's display capabilities, using rules in the device repository. The color depth and image type combinations supported for output images are shown in *ICS image formats*. For conversion rules see *ICS transcoding rules*.

Image optimization

When the color gamut has to be reduced from the input to the output images, the algorithms used are configurable to the extent available.

Different palette selection strategies and dithering algorithms typically provide different levels of visual quality, but exhibit trade-offs against speed and image size. A choice of palette selection algorithms and dithering algorithms/levels is provided. The algorithms used are configurable on a per-server basis but can be overridden via URL parameters.

If too small restriction is given for a given output format (there tends to be a fixed file content overhead for image file formats below which sizes can't be generated) then the size restriction cannot be applied. It is likely in this case that the smallest possible image will be generated (for example 1x1 pixel black and white).

Image scaling

The dimensions of the output image are scaled to the size required for the device.

- Images are scaled based on width and height. No change in the aspect ratio occurs.
- The scaling algorithm minimizes scaling artifacts
- By default, the transcoder does not make images larger than the source image. However, this setting is configurable and is capable of being overridden with URL parameters.
- In the Eclipse interface, users can minimize the effects of scaling wide images by specifying the left and right width limits of an area to preserve. Vertical cropping is not supported
- Format preservation, if the image is in a format supported by a device, and the size matches the requested size.

Image file size restriction

The transcoding engine is able to limit the output file to a size that can be handled by the device. This parameter is obtained from the device repository and can be provided via the request URL.

The file size restriction has the following properties:

- For file formats that use lossless or no compression, the file size reduction strategy is to shrink the pixel dimensions of the image. This applies to true color PNG, WBMP and BMP output.
- For file formats that use lossy, variable compression, the initial file size reduction strategy is to increase the compression factor. However, there is a maximum compression threshold, above which pixel dimension reduction is used. This maximum threshold has a default value that is configurable at the server level, but this value is capable of being overridden via a URL parameter. This applies to JFIF/JPEG output.
- For indexed (palette-based) file formats, the initial strategy is to reduce the palette size and therefore the effective bit depth of the image. However, there is a minimum palleted size threshold, below which pixel dimension reduction is used instead. This minimum threshold has a default value that is configurable at the server level, but this value is capable of being overridden via a URL parameter. This applies to indexed PNG output.

ICS image formats

The table shows the image input and output formats supported by ICS. The cells containing an 'X' in the Input and Output columns indicate that the format is supported.

Format	Input	Output	Notes
Windows BMP	X		Monochrome, indexed, true color
GIF87a	X		Monochrome, greyscale, indexed
GIF89a including animation	X	X	RLE or LZW compression
JFIF/JPEG	X	X	Baseline and progressive
PNG	X	X	Greyscale, indexed and true color
TIFF	X	X	LZW compression is not supported
SVG	X		JFIF/JPEG, PNG and WBMP output
WBMP	X	X	Monochrome only

Note: In the `ics-config.xml` file, you can configure ICS to use LZW compression for GIF 89a output by setting the `gifSupport` attribute on the `image` element to 'true'.

ICS transcoding rules

The ICS servlet applies transcoding rules using device policies to determine the output format of the transcoded image for a device. The table shows the names and descriptions of the rules used in ICS native mode. These rules are defined in the device repository distributed with MCS.

Note: Custom device policy transcoding rules are not supported by ICS.

Rule Name	Description
gp1	1 Bit (B and W) greyscale PNG
gp2	2 Bit greyscale PNG
gp4	4 Bit greyscale PNG
gp8	8 Bit greyscale PNG
gp16	16 Bit greyscale PNG
gw1	1 Bit (B and W) greyscale WBMP
gj8	8 Bit greyscale JPEG
cb24	24 Bit color BMP
cp8	8 Bit color PNG
cp24	24 Bit color PNG
cj24	24 Bit color JPEG
cg8	8 Bit color GIF
gg1	1 Bit (B and W) GIF
gg2	2 Bit greyscale GIF
gg4	4 Bit greyscale GIF
gg8	8 Bit greyscale GIF
ct24	24 Bit color TIFF

ICS configuration parameters

The table describes the options and shows the override settings that you can use in ICS native and TransForce compatibility modes. The parameters are expressed as attributes in the configuration file.

Parameter	Description	Native	TransForce
serverMode	Defines the ICS mode (native, transforce)		
imgHost	Host name of the image server	v.imgHost	tf.source.host
imgPort	Port of the image server	v.imgPort	tf.source.port
proxyHost	Proxy server to use for retrieving remote images		
proxyPort	Port that the proxy server is listening on		
maxRedirects	The number of times to retry following a redirect response		
scaleMode	Method of scaling (bilinear, bicubic)	v.scaleMode	
scaleLarger	Specifies if an image can be scaled up (true, false)	v.scaleLarger	
qualityMin	Minimum lossy compression quality before ICS uses pixel reduction. Values range from 0 to 100 (no loss) - default 90. Ignored if `v.maxSize` or `tf.maxfilesize` are not specified.	v.qualityMin	
paletteMin	Minimum allowable palette size with lossy compression before ICS uses pixel reduction. The value may be 1, 2, 4 or 8 (default) but cannot exceed bit depth of the source. Ignored if `v.maxSize` or `tf.maxfilesize` are not specified.	v.paletteMin	
dither	Dither algorithms used to quantize an image to a lower target bit size than the source (ordered-dither, floyd-steinberg, jarvis, stucki)	v.ditherMode	
gifSupport	Specifies if GIF is supported as an input image type.		
jpegFormat	Specifies the mode in which a JPEG image will be rendered (baseline, progressive)	v.jpegFormat	
watermarkURL	Specifies the URL of a watermark to overlay	v.watermarkURL	
deviceRepositryURL	Specifies the URL of the device repository to use when evaluating the need to preserve image formats	v.deviceRepositryURL	

Overriding configuration settings

When the ICS servlet is running in native mode it supports features that allow you to override several aspects of the transcoding. When the ICS runs in TransForce compatibility mode it supports a limited subset of the facilities supported by TransForce.

URL format

The URL format is made up of the host and port of the server, the transcoding rule, the relative path of the image and any additional transcoding parameters.

`http://myserver.com/images/cp8/image.jpg?v.imgHost=www.somesite.com&amp;v.imgPort=80`
This will result in ICS fetching the image from http://www.somesite.com:80/images/image.jpg.

Image format and color depth

The output image is encoded in the format specified by `[transcoding_rule]` from the URL (see *ICS transcoding rules*).

All conversion of format and color depth occur after the image has been scaled. This ensures optimal quantizing of colors as well as improvements in performance.

In native mode you have some control over the output of JPEG and PNG images. You can specify the exact JPEG format, and the dithering algorithm that ICS should use with the `v.jpegFormat` and `v.ditherMode` parameters.

There are no comparable overrides for TransForce mode.

Image scaling

You can specify the scaling of the output image. All scaling maintains the aspect ratio of the source.

The `v.width` or `tf.width` parameters specify the width in pixels for the output image. If `scaleLarger` or `v.scaleLarger` is set to "false" the width will not be allowed to exceed the width of the input image. If it does then it will be set to the width of the input image. The width may be less than the given value if the height parameter restricts the image size to a greater extent.

The `v.height` or `tf.height` parameters specify the height in pixels for the output image. The effect is to set the height of the image to be no larger than the parameter value. Height may be less than the given value if the width parameter restricts the image size.

Note: You can specify width alone, but if you define a height value, you must also provide a width setting.

`v.scaleMode` is available to override the scaling algorithm.

Image cropping to avoid excessive width scaling can also be specified using the `v.p` parameter in the form `v.p=[[m],n]`, where 'm' and 'n' are the positions of the first and last pixel columns of the area that should be preserved. The columns are counted from the left of the image and start at 0.

File size restriction

For mobile devices which may have limited memory, or restrictions on the size of image files, you can control the size of the output image.

ICS will try to use lossy compression or palette reduction on an image for those output formats that support it. If this fails to yield a small enough image, or the image format does not support lossy reduction, ICS uses pixel reduction.

`v.maxSize` and `tf.maxfilesize` specify the maximum allowable size of the image in bytes. You can also use the overrides `v.qualityMin` and `v.paletteMin`.

References

Apache Software Foundation
    `http://apache.org`
Free Software Foundation
    `http://www.fsf.org`
Jaxen Project
    `http://jaxen.org/`
Sun Microsystems Inc
    `http://www.sun.com`
Java Advance Imaging API
    `http://java.sun.com/products/java-media/jai/index.jsp`