

Sun Certified Mobile Application Developer (311-110)
Sun Microsystems, Inc.

Exam Notes
Sathya Srinivasan

02/11/2004

Table of Contents

Push Registry Benefits	3
Push Registry Limitations	3
Use of Push Registry in applications.....	4
Discovery of Push Registry	5
Dynamic vs. Static Push Registry declarations	5
Static Push Registry declaration	5
Dynamic Push Registry declaration	5
Push Registry connection types	5

Chapter 7 Push Registry

Explain MIDP 2.0 Push Registry benefits, and limitations, and describe its use in applications.

Push Registry Benefits

- Push Registry can be used to start an application based on an event received from a port that it is registered to.
 - If the application is not running or is paused when an event is received, the Application Management Software (AMS) will start the application in the standard process. The application can then receive the message or stream using the standard process.
 - If the application is already running, nothing happens since the application will already be listening for messages or streams anyway.
- Push Registry can be used for `socket`, `datagram`, `sms`, and `cbs` protocols.
- IPv4 and IPv6 URLs are supported.
- If an application is not running, the messages will be received by the AMS and delivered to the application after starting the application for `Message` connections.
- If an application is not running, the stream will be preserved by the AMS and handed over to the application after starting it for `datagram` or `socket` connections, as long as the stream does not timeout.

Push Registry Limitations

- Two MIDlet suites CANNOT share the same Push Registry connection.
- If all the statically declared connections could not be registered successfully during installation of the application or if there are errors in the declaration, the application WILL NOT be installed.
- Not all implementations are required to support Push Registry for protocols that typically support Push Registry. In such cases, attempts at registration will result in a `ConnectionNotFoundException`. For example, the system might support `socket` protocol, but not necessarily Push Registry for the `socket` protocol.

Use of Push Registry in applications

- When a Push Registry event is received by the AMS, what happens to the currently running application (it cannot be the application to which the event is registered to, since Push Registry will not listen to events of the currently running application) is upto the implementation. It MAY pause or destroy the running application before starting the target application.
- If an application is running and has registered itself to the Push Registry for some events
 - The AMS will NOT listen for events for that application.
 - If the application calls the `Connection.close()` method, then any events sent to that port will be lost (the Push Registry will not be listening as the app. is currently running).
 - Once the application is destroyed, the AMS will start listening for events for that application.
- An application can register itself for an alarm using the `registerAlarm(String class, long time)`.
 - The `class` should be a MIDlet in the currently running MIDlet suite which is to be notified.
 - Upon receipt of an event, the `class` will be notified after the time expires.
 - At most, only one alarm can be registered per registry entry.
 - The MIDlet must be registered in the descriptor or the manifest file.

Develop applications that correctly use MIDP 2.0 Push Registry including discovery, dynamic vs. static, and recognizing the types of connections that can and cannot be accepted.

Discovery of Push Registry

- The Push Registry and the AMS is responsible only for starting a non-running application when an event that the non-running application has registered itself for in the Push Registry is received by the device. It is the application's responsibility to retrieve the message from the connection.
- See [Use of Push Registry in applications](#) for more details.

Dynamic vs. Static Push Registry declarations

Static Push Registry declaration

The following attributes MUST be present in the Application Descriptor (JAD) or the manifest file of the JAR of the application.

Attribute	Value Format
MIDlet-Push-<n>	<p><Connection-URL>, <MIDletClassName>, <AllowedSender></p> <p>where</p> <p>n is a sequentially incrementing number from 1 with no breaks.</p> <p>Connection-URL is the URL to listen to of the form <code>protocol://:port</code></p> <p>MIDletClassName is the class registered to receive the event. This should be declared in one of the MIDlet-<n> attributes.</p> <p>AllowedSender is the address or address range of the senders permitted to send events. This is also called as a filter.</p>

- If there is a break in the sequence number, only the contiguously numbered entries (starting from 1) are considered. The rest are ignored.
- If there is an error in the declaration, then the application will NOT be installed.
- If the requested port is reserved or is restricted, then the application will NOT be installed.
- If the MIDletClassName is not declared in one of the MIDlet-<n> attributes, then the application will NOT be installed.

Dynamic Push Registry declaration

- An application can be registered dynamically to the Push Registry by invoking the static `PushRegistry.registerConnection(String connection, String midlet, String filter)` method.
 - The parameters are the same as that of the static declaration.

Push Registry connection types

See [Dynamic vs. Static Push Registry declarations](#).