



SMS Proxy User's Guide

Version 1.0

CONFIDENTIAL INFORMATION

The information herein is the property of Ex Libris Ltd. or its affiliates and any misuse or abuse will result in economic loss. DO NOT COPY UNLESS YOU HAVE BEEN GIVEN SPECIFIC WRITTEN AUTHORIZATION FROM EX LIBRIS LTD.

This document is provided for limited and restricted purposes in accordance with a binding contract with Ex Libris Ltd. or an affiliate. The information herein includes trade secrets and is confidential.

DISCLAIMER

The information in this document will be subject to periodic change and updating. Please confirm that you have the most current documentation. There are no warranties of any kind, express or implied, provided in this documentation, other than those expressly agreed upon in the applicable Ex Libris contract. This information is provided AS IS. Unless otherwise agreed, Ex Libris shall not be liable for any damages for use of this document, including, without limitation, consequential, punitive, indirect or direct damages.

Any references in this document to third-party material (including third-party Web sites) are provided for convenience only and do not in any manner serve as an endorsement of that third-party material or those Web sites. The third-party materials are not part of the materials for this Ex Libris product and Ex Libris has no liability for such materials.

TRADEMARKS

"Ex Libris," the Ex Libris bridge , Primo, Aleph, Alephino, Voyager, SFX, MetaLib, Verde, DigiTool, Preservation, URM, Voyager, ENCompass, Endeavor eZConnect, WebVoyage, Citation Server, LinkFinder and LinkFinder Plus, and other marks are trademarks or registered trademarks of Ex Libris Ltd. or its affiliates.

The absence of a name or logo in this list does not constitute a waiver of any and all intellectual property rights that Ex Libris Ltd. or its affiliates have established in any of its products, features, or service names or logos.

Trademarks of various third-party products, which may include the following, are referenced in this documentation. Ex Libris does not claim any rights in these trademarks. Use of these marks does not imply endorsement by Ex Libris of these third-party products, or endorsement by these third parties of Ex Libris products.

Oracle is a registered trademark of Oracle Corporation.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Ltd.

Microsoft, the Microsoft logo, MS, MS-DOS, Microsoft PowerPoint, Visual Basic, Visual C++, Win32, Microsoft Windows, the Windows logo, Microsoft Notepad, Microsoft Windows Explorer, Microsoft Internet Explorer, and Windows NT are registered trademarks and ActiveX is a trademark of the Microsoft Corporation in the United States and/or other countries.

Unicode and the Unicode logo are registered trademarks of Unicode, Inc.

Google is a registered trademark of Google, Inc.

Copyright Ex Libris Limited, 2011. All rights reserved.

Document updated: February 15, 2011

Web address: <http://www.exlibrisgroup.com>

Table of Contents

1	About This Guide	4
2	Overview of the SMS Proxy	5
	Terms and Definitions	6
3	SMS Proxy Configuration	7
4	SMS Proxy Workflow	9
	The SMS Proxy APIs	10
5	Adding Support for Additional SMS Providers	11

1

About This Guide

This guide explains how to use the SMS proxy in conjunction with the following Ex Libris products:

- Aleph (version 19.01 and later)
- Primo (version 2.0 and later)
- Voyager (version 8.0 and later)

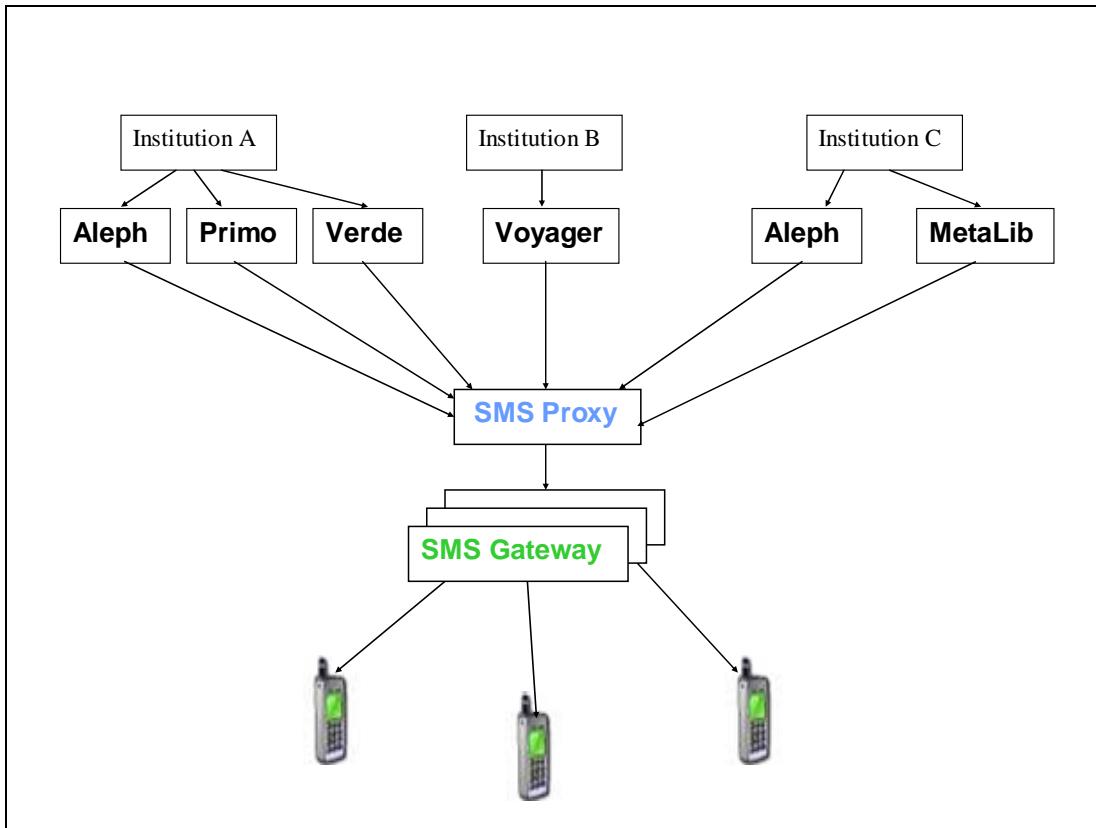
2

Overview of the SMS Proxy

The SMS proxy is a Java .war application running on a JBoss server. It is used to pass SMS messages to providers who specialize in sending mass amounts of text messages over cellular networks. Since each provider has a different method (API) of receiving the requests, the main functionality of the SMS proxy is to convert requests it receives to the appropriate format as defined by the providers.

The SMS proxy serves as an intermediary. Instead of contacting the provider directly, the main application (for example, Aleph, Voyager, or Primo) contacts the SMS proxy with X-services, and the SMS proxy contacts the provider according to its defined methods. See [The SMS Proxy APIs](#) for more information.

One SMS proxy installation can serve different applications at several institutions, as illustrated in the following image:



Terms and Definitions

- **Provider** – A company such as SimpleWire or Clickatell that provides SMS gateways. The provider accepts requests for SMS messages (usually over the Internet) and forwards them to a cellular network.
- **Adapter** – A component inside the SMS proxy that is responsible for contacting a specific provider.

3

SMS Proxy Configuration

The SMS proxy uses a single configuration file called `SmsProxyConf.xml` that includes all of the component parameters. It is located in the `./system/conf` directory, under the server JBoss HOME. The JBoss Home is the base directory of JBoss distribution. For example, in Aleph this could be: `$aleph_dev/ng/aleph/home`. The `SmsProxyConf.xml` configuration file contains the following lists:

- **IP address** – The SMS proxy only accepts requests arriving from these addresses.
- **Institutions** – Each institution has a code, a name, and a `ProviderCode` that identifies the provider it uses. Each institution can have only one provider. Since two institutions can use the same provider, but with different user names and passwords, each institution has fields for the user name and password as well.
- **Providers** – Each provider has a code (which connects it to an institution), a name, and a component, which is the name of the Java class that is used to handle the connection to the provider.

```
<?xml version="1.0" encoding="UTF-8"?>
<SmsProxyRoot xmlns="http://com/exlibris/core/sms/proxy/conf">

    <Senders>
        <Sender>127.0.0.1</Sender>
        <Sender>10.1.234.78</Sender>
    </Senders>

    <Institutions>
        <Institution name="Exlibris Demo Library" code="USM50">
            <ProviderCode>simplewire</ProviderCode>
            <ProviderUser>415-562-401-111111</ProviderUser>
            <ProviderPassword>123456</ProviderPassword>
        </Institution>
        <Institution name="Another Demo Library" code="USM51">
```

```
<ProviderCode>simplewire</ProviderCode>
<ProviderUser>415-562-401-22222</ProviderUser>
<ProviderPassword>6543221</ProviderPassword>
</Institution>
<Institution name="Clickatell Demo Library" code="CLI50">
    <ProviderCode>clickatell</ProviderCode>
    <ProviderUser>uuuunnnn</ProviderUser>
    <ProviderPassword>qwerty:123456</ProviderPassword>
</Institution>
</Institutions>

<Providers>
    <Provider name="SimpleWire" code="simplewire">
        <component>com.exlibris.core.sms.proxy.SimpleWireProvider</component>
    </Provider>
    <Provider name="Clickatell" code="clickatell">
        <component>com.exlibris.core.sms.proxy.ClickatellProvider</component>
    </Provider>
</Providers>

</SmsProxyRoot>
```

4

SMS Proxy Workflow

The SMS proxy operates according to the following workflow:

- 1 The SMS proxy is invoked via HTTP GET requests. The HTTP GET URL should have the prefix `core-sms-proxy/sms?` followed by the request parameters. For example: `http://localhost:1801/core-sms-proxy/sms?&action=submit&institution=CLI50&phone=972509170097 &message=see%20you%20there`
- 2 The SMS proxy checks the list of senders to confirm that the request arrived from a valid IP address. After this is confirmed, the institution name is located in the configuration file, and the `ProviderCode` is used to locate the configuration for the provider.
- 3 The parameters from the URL, together with the user name and password from the configuration file, are used to contact the provider via the methods in the Java class written for this provider.
- 4 The SMS proxy analyzes the response it receives from the provider and returns an XML with information about the success or failure of the operation.

Note that a successful response from the provider does not indicate that the SMS message was actually received by the recipient cellular phone. It only indicates that it was received by the provider, and it will be sent after processing. The duration of the processing can vary between a few seconds and several hours.

The returned XML might also include a message ID that identifies the specific SMS message as received by the provider. The message ID can be used to check the status of the message (if it was received by the recipient cellular phone or if it is still queued) using the **GetStatus** API.

The SMS Proxy APIs

Aside from the main API (action=submit) that was described above, the SMS proxy has three more APIs:

- **GetStatus** – Queries the provider regarding the status of an SMS message that was sent using the submit API
- **UpdateParam** – Updates the configuration file
- **LoadConfig** – Reads the configuration file (after changing it manually)

Updating the configuration can be done in two ways:

- Using the **UpdateParam** API.
- Manually, with a text editor. After saving the file, the JBoss server needs to be restarted for the configuration to be read again. Alternatively, the **LoadConfig** API can be used instead.

5

Adding Support for Additional SMS Providers

By default, the SMS proxy supplies two adapters. Each adapter supports sending SMS messages via one of the following providers:

- SimpleWire www.openmarket.com
- Clickatell www.clickatell.com

The architecture of SMS proxy is designed to allow for adapters written by customers and, thereby, enable support for providers that institutions are already using.

The code should be written in the Java programming language and deployed on the JBoss as a .jar file.

The configuration should be updated with the name of the Java class file by adding a new provider record in the `SmsProxyConf.xml` file.

For more information, refer to the Developer Zone on the EL Commons collaborative Web site (<http://www.exlibrisgroup.org>).