

中国移动代理服务器 MAS V2.0

WebService 接口插件

开发手册



深圳市嘉讯软件有限公司

2008 年 9 月

目录

1. 引言	4
1.1. 编写目的	4
1.2. 相关术语与缩略语解释	4
2. 概述	4
3. 二次开发环境	4
4. 开发步骤	5
4.1. 在 Mas 服务器新建一个企业应用	5
4.2. 启动企业应用	6
4.3. 访问并获取 Webservice 方式通信适配插件服务地址	6
4.4. 客户端及服务端代码生成	6
4.5. 接口开发及使用举例	6
4.5.1. 短信接口	6
4.5.1.1. 发送短信-sendSms 操作	6
4.5.1.2. 获取短信发送状态-getSmsDeliveryStatus 操作	8
4.5.1.3. 通知短信发送状态-notifySmsDeliveryStatus 操作	8
4.5.1.4. 获取 MO 短信-getReceivedSms 操作	9
4.5.1.5. 通知 MO 短信到达-notifySmsReception 操作	10
4.5.2. 彩信接口	10
4.5.2.1. 发送彩信-sendMessage 操作	10
4.5.2.2. 获取彩信发送状态-getMessageDeliveryStatus 操作	11
4.5.2.3. 通知彩信发送状态-notifyMessageDeliveryReceipt 操作	12
4.5.2.4. 获取 MO 彩信摘要-getReceivedMessages 操作	12
4.5.2.5. 获取 MO 彩信-getMessage 操作	13
4.5.2.6. 通知 MO 彩信到达-notifyMessageReception 操作	14
4.5.3. Wappush 短信接口	14
4.5.3.1. 发送 Wappush-sendPush 操作	14
4.5.3.2. 获取 Wappush 发送状态-getPushDeliveryStatus 操作	16
4.5.3.3. 通知 Wappush 发送状态-notifyPushDeliveryReceipt 操作	16
4.6. 常见错误及处理方式	17
5. 接口描述	17
5.1. 短信接口	17
5.1.1. 接口说明	17
5.1.2. 流程说明	18
5.1.3. 数据类型	19
5.2. 彩信接口	21
5.2.1. 接口说明	21
5.2.2. 流程说明	21
5.2.3. 数据类型	22

5.3.	Wappush 短信接口	23
5.3.1.	接口说明	23
5.3.2.	流程说明	23
5.3.3.	数据类型	24
6.	参考文档	24

1. 引言

1.1. 编写目的

本文档描述了 MAS2.0 基座 WebService 通信接口的使用方法，文档的使用对象为 MAS 开发人员、数字化部队、SI 工程技术人员、SI 开发人员、集团客户技术人员。

本文档的读者需要有一定的 WebService 相关知识。

1.2. 相关术语与缩略语解释

缩写词	英文解析	中文解析
MAS	Mobile Agent Server	移动代理服务器
HTTP	Hypertext Transfer Protocol	超文本传输协议
HTTPS	Secure HTTP	加密的 HTTP 协议
SOAP	Simple Object Access Protocol	简单对象访问协议
UCA	Unified Communication Agent	统一通信代理
WS	WebService	Web 服务接口

2. 概述

开发环境 WebService 方式通信提供了对各种通信能力的统一封装，供集团客户应用系统和 MAS 服务器应用插件进行调用。同时，提供统一 WebService 方式封装也降低了 SI 的开发难度，将繁杂的通信能力协议与具体的业务应用分离开来，对 SI 呈现简单的标准接口。

WebService 方式通信适配插件接口封装了以下移动通信能力：

- 短信
- 彩信
- wappush
- ussd(暂不可用)
- LBS(暂不可用)

此外，还提供基座与应用插件之间的插件管理功能接口。

MAS 服务器应提供配置工具，指定对于某个集团客户应用系统或应用插件，通过 WebService 方式通信适配插件或数据库方式通信适配插件接入 MAS 服务器。

3. 二次开发环境

- MAS2.X 运行环境
- MAS 服务平台环境
- WebService 适配插件开关打开


4.1. 在 Mas 服务器新建一个企业应用

- 两者的区别是，如果是插件在使用通信能力之前必须向 MAS 服务器注册，而如果开发的是集团客户应用，就不需预先向 MAS 服务器注册。

- 在开发完成后准备调试前，先运行 MAS2.0 服务器，然后在插件管理里新建一个应用或插件。

应用类型：	<input checked="" type="radio"/> 企业应用 <input type="radio"/> 应用插件		
显示类别：	内部信息		
标识 *：	公文到达提醒		
名称 *：	公文到达提醒		
通信接口方式	WebService		
类型：	INDUSTRY		
应用IP地址 *：	192.168.0.238		
短信应用：	<input checked="" type="radio"/> 是 <input type="radio"/> 否 扩展号码 *：122 匹配格式 <input type="radio"/> 精确 <input checked="" type="radio"/> 模糊 MO命令字 *：122 (多个用;号隔开) 业务代码：122		
彩信应用：	<input type="radio"/> 是 <input checked="" type="radio"/> 否		
WAP应用：	<input type="radio"/> 是 <input checked="" type="radio"/> 否		
USSD应用：	<input type="radio"/> 是 <input checked="" type="radio"/> 否		
LBS应用：	<input type="radio"/> 是 <input checked="" type="radio"/> 否		
CPU门限值：	警戒 0.5	严重 0.7	高危 0.9
内存门限值：	警戒 0.5	严重 0.7	高危 0.9
硬盘门限值：	警戒 0.5	严重 0.7	高危 0.9

图 5-1 新增插件页面

增加 编辑 删除 插件监控 查询 导入 类别管理 升级 卸载												
	标识	名称	类型	通信接口	短信应用	彩信应用	WAP应用	USSD应用	LBS应用	状态	开关	安装
<input type="checkbox"/>	公文到达提醒	公文到达提醒	INDUSTRY	WebService	短信子号: 122 MO命令字: 122	否	否	否	否	需要注册	运行	否
<input type="checkbox"/>	P0200000000000121	P0200000000000121	INDUSTRY	Database	短信子号: 121 MO命令字: 121	否	否	否	否	正常	暂停	否
<input type="checkbox"/>	api3	api3	ERP	InnerSpecific	短信子号: 105 MO命令字: 105	彩信子号: 105	是	是	是	正常	暂停	否
<input type="checkbox"/>	api2	api2	ERP	InnerSpecific	短信子号: 104 MO命令字: 104	彩信子号: 104	是	是	是	正常	暂停	否
<input type="checkbox"/>	api	api	ERP	InnerSpecific	短信子号: 103 MO命令字: 103	彩信子号: 103	是	是	是	正常	暂停	否
<input type="checkbox"/>	P0200000000000100	默认插件	INDUSTRY	InnerSpecific	短信子号: 00000 MO命令字: DEFAULTMO	彩信子号:	是	是	是	正常	暂停	

首页 上页 下页 尾页 第 1/1 页 共 6 条 1

图 5-2 插件管理页面

注意：新建时“通信接口方式”选择“WebService”。

这里新建的应用或插件中的“标识”即为以下开发代码时所设置的“应用 ID 或插件的 ID”。

4.2. 启动企业应用

要使新建的插件 Webservice 接口可用，需要点击“开关”列的“运行”按钮，如图 5-2 中的“开关”列，使开关状态处于运行状态。否则在使用 WS 接口通信时可能会报 POL0906 错误。当然要确保 Webservice 接口能正常使用，还需要管理平台相应的业务能力。

4.3. 访问并获取 Webservice 方式通信适配插件服务地址

在 IE 浏览器中输入如 http://IP/Mas2/services/cmcc_mas_wbs 地址，其中‘IP’为 Mas 服务器的 IP 地址，‘Mas2’为 Mas 的工程名，‘/services/cmcc_mas_wbs’为服务地址名，若访问服务地址成功，在 IE 中将显示如图 5-3 所示信息。

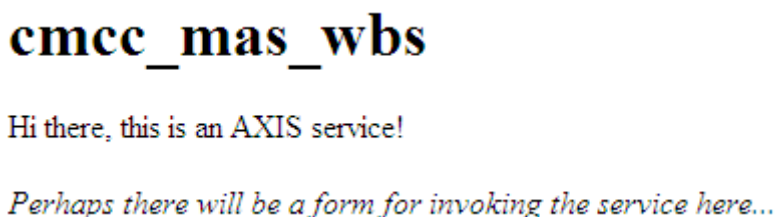


图 5-3 Webservice 服务访问显示信息

正确显示服务信息后，记住此服务地址：http://IP/Mas2/services/cmcc_mas_wbs，此地址即为‘WebService 方式通信适配插件服务地址’。

4.4. 客户端及服务端代码生成

在开发 Webservice 接口插件前，首先需要生成其客户端和服务端代码。用户可使用自己的生成方式生成代码，其中一种生成方式如下：

首先获取 WSDL 地址。在‘WebService 方式通信适配插件服务地址’后添加‘?wsdl’，然后在 IE 中输入此地址，如 http://IP/Mas2/services/cmcc_mas_wbs?wsdl，访问可显示 WSDL 信息。

利用 WSDL 地址，可使用 Axis 提供的 WSDL2Java 命令行工具 `org.apache.axis.wsdl.WSDL2Java`，可用此工具在命令行生成其客户端和服务端代码。将生成的客户端及服务端代码部署到开发工程中，然后进行后续的开发步骤。

4.5. 接口开发及使用举例

4.5.1. 短信接口

4.5.1.1. 发送短信-sendSms 操作

4.5.1.1.1. 操作说明

sendSms 调用是由应用系统或插件发起，请求发送一条短消息到一个指定地址（或地址集合），该短消息由 Message 描述，目的地址由 DestinationAddress 描述。

4.5.1.1.2. 开发代码举例

输入：sendSmsRequest

参数	类型	可选性	描述
ApplicationID	xsd:string	M	应用 ID 或插件的 ID。
DestinationAddresses	xsd:anyURI [0..unbounded]	M	短消息要被发送到的地址。 群发短消息的最大数量为 254。
ExtendCode	xsd:string	O	指由该应用填写的内部扩展号码。 MAS 服务器需自动补充为此业务分配的长服务号码。
Message	xsd: string	M	在短消息中发送的文本。
MessageFormat	MessageFormat	M	消息编码类型。
SendMethod	SendMethodType	M	发送消息选项。
DeliveryResultRequest	xsd:boolean	O	指示是否需要网络侧返回递交状态报告。若无，则不返回。True 表示需要网络侧返回递交状态报告，false 表示不需要网络侧返回递交状态报告。

输出：sendSmsResponse

参数	类型	可选性	描述
RequestIdentifier	xsd:string	M	标识一个特定的短消息发送请求。

代码实现如下：

```

Cmcc_mas_wbs_ServiceLocator      cmcc_mas_wbs_ServiceLocator      =      new
Cmcc_mas_wbs_ServiceLocator();
// 设置 Webservice 方式通信适配插件服务地址
cmcc_mas_wbs_ServiceLocator.setcmcc_mas_wbsEndpointAddress("WebService 方式通信适配插件服务地址");
// 生成客户端访问接口
Cmcc_mas_wbs_PortType            cmcc_mas_wbs_PortType            =
cmcc_mas_wbs_ServiceLocator.getcmcc_mas_wbs();

// 应用系统或插件请求发送的短消息
SendSmsRequest sendSmsRequest = new SendSmsRequest();
// 设置应用 ID 或插件的 ID
sendSmsRequest.setApplicationID(applicationID);
// 设置指示是否需要网络侧返回递交状态报告。若无，则不返回。True 表示需要网络侧返回递交状态报告，false 表示不需要网络侧返回递交状态报告
sendSmsRequest.setDeliveryResultRequest(deliveryResultRequest);
// 设置短消息要被发送到的地址。群发短消息的最大数量为 254
sendSmsRequest.setDestinationAddresses(destinationAddresses);
// 设置由该应用填写的内部扩展号码。MAS 服务器需自动补充为此业务分配的长服务号码
sendSmsRequest.setExtendCode(extendCode);
// 设置在短消息中发送的文本
sendSmsRequest.setMessage(message);
// 设置消息编码类

```

```

sendSmsRequest.setMessageFormat(messageFormat);
// 设置发送消息选项
sendSmsRequest.setSendMethod(sendMethod);
// 发送短消息（返回发送短消息响应）
SendSmsResponse                                sendSmsResponse                                =
cmcc_mas_wbs_PortType.sendSms(sendSmsRequest);

```

4.5.1.2. 获取短信发送状态-getSmsDeliveryStatus 操作

4.5.1.2.1. 操作说明

用于集团客户应用系统或应用插件查询短消息的发送状态。

4.5.1.2.2. 开发代码举例

输入：GetSmsDeliveryStatusRequest

参数	类型	可选性	描述
ApplicationID	xsd:string	M	应用 ID 或插件 ID。
RequestIdentifier	xsd:string	M	在发送短信时由 WebService 方式通信适配插件分配的发送请求标识。

输出：GetSmsDeliveryStatusResponse

参数	类型	可选性	描述
DeliveryStatus	DeliveryInformation [0..unbounded]	M	列出短消息发送状态。

代码实现如下：

```

Cmcc_mas_wbs_ServiceLocator    cmcc_mas_wbs_ServiceLocator    =    new
Cmcc_mas_wbs_ServiceLocator();
// 设置 WebService 方式通信适配插件服务地址
cmcc_mas_wbs_ServiceLocator.setcmcc_mas_wbsEndpointAddress("WebService 方式通
信适配插件服务地址");
// 生成客户端访问接口
Cmcc_mas_wbs_PortType          cmcc_mas_wbs_PortType          =
cmcc_mas_wbs_ServiceLocator.getcmcc_mas_wbs();

// 查询短消息发送状态的请求
GetSmsDeliveryStatusRequest    getSmsDeliveryStatusRequest    =    new
GetSmsDeliveryStatusRequest();
// 设置应用 ID 或插件 ID
getSmsDeliveryStatusRequest.setApplicationID(applicationID);
// 设置在发送短信时由 WebService 方式通信适配插件分配的发送请求标识
getSmsDeliveryStatusRequest.setRequestIdentifier(requestIdentifier);
// 查询发送状态（返回发送状态信息数组）
DeliveryInformation[]          deliveryInformation              =
cmcc_mas_wbs_PortType.getSmsDeliveryStatus(getSmsDeliveryStatusRequest);

```

4.5.1.3. 通知短信发送状态-notifySmsDeliveryStatus 操作

4.5.1.3.1. 操作说明

应用系统或应用插件通过通信适配插件发送了短消息之后,通信适配插件对该短消息的发送分配了一个标识符 RequestIdentifier。当通信适配插件获得状态报告后,调用本接口将短消息的发送状态报告给应用侧,前提是应用系统或插件具备 Web 服务端。状态信息在 DeliveryInformation 中具体描述。

4.5.1.3.2. 开发代码举例

输入: notifySmsDeliveryStatusRequest

参数	类型	可选性	描述
RequestIdentifier	xsd:string	M	标识一个短消息发送请求。
DeliveryInformation	deliveryInformation[0..unbounded]	M	短消息递交状态参数。

输出: notifySmsDeliveryStatusResponse

4.5.1.4. 获取 MO 短信-getReceivedSms 操作

4.5.1.4.1. 操作说明

用于集团客户应用系统和应用插件从通信适配插件获取接收到的短消息。通信适配插件返回 receivedSms, 包含短消息的发送者和内容。

4.5.1.4.2. 开发代码举例

输入: getReceivedSmsRequest

参数	类型	可选性	描述
ApplicationID	xsd:string	M	应用 ID 或插件 ID。

输出: GetReceivedSmsResponse

参数	类型	可选性	描述
ReceivedSms	SmsMessage[0..unbounded]	M	接收的短消息

代码实现如下:

```

Cmcc_mas_wbs_ServiceLocator cmcc_mas_wbs_ServiceLocator = new
Cmcc_mas_wbs_ServiceLocator();
// 设置 Webservice 方式通信适配插件服务地址
cmcc_mas_wbs_ServiceLocator.setcmcc_mas_wbsEndpointAddress("WebService 方式通信适配插件服务地址");
// 生成客户端访问接口
Cmcc_mas_wbs_PortType cmcc_mas_wbs_PortType =
cmcc_mas_wbs_ServiceLocator.getcmcc_mas_wbs();

// 获取接收到的短消息的请求
GetReceivedSmsRequest getReceivedSmsRequest = new GetReceivedSmsRequest();
// 设置应用 ID 或插件 ID
getReceivedSmsRequest.setApplicationID(applicationID);

```

```
// 获取接收到的短消息
```

```
cmcc_mas_wbs_PortType.getReceivedSms(getReceivedSmsRequest);
```

4.5.1.5. 通知 M0 短信到达-notifySmsReception 操作

4.5.1.5.1. 操作说明

通信适配插件收到发送到特定地址的短消息时，调用本接口，通知集团客户应用系统和应用插件有短消息到达，前提是应用系统或插件具备 Web 服务端。

4.5.1.5.2. 开发代码举例

输入：notifySmsReceptionRequest

参数	类型	可选性	描述
Message	SmsMessage	M	在短消息中发送的文本。

输出：notifySmsReceptionResponse

4.5.2. 彩信接口

4.5.2.1. 发送彩信-sendMessage 操作

4.5.2.1.1. 操作说明

用于集团客户应用系统和应用插件向通信适配插件提交发送彩信（包括标题和内容）请求，并指明是否需要网络侧返回状态报告。本操作支持群发。在响应中，通信适配插件返回对本次请求的标识。

4.5.2.1.2. 开发代码举例

输入：sendMessageRequest

参数	类型	可选性	描述
ApplicationID	xsd:string	M	应用ID或插件ID
addresses	xsd:anyURI [0..unbounded]	M	消息要被发送到的地址。
ExtendCode	xsd:string	O	指由该应用填写的应用内部扩展号码。MAS 服务器需自动补充为此业务分配的服务号码。
Subject	xsd:string	O	本参数指示消息的主题。
Priority	MessagePriority	O	本参数代表消息的优先级。如果未指明，按缺省优先级处理。
receiptRequest	xsd: boolean	O	指示是否需要返回递交状态报告。true 表示需要，false 表示不需要。若无，则不返回。
Content	xsd: string	O	和消息一起发送的数据。

输出：sendMessageResponse

参数	类型	可选性	描述
requestIdentifier	xsd:string	M	标识本次发送请求，用于 getMessageDeliveryStatus 调用。

代码实现如下：

```
Cmcc_mas_wbs_ServiceLocator cmcc_mas_wbs_ServiceLocator = new
```

```

Cmcc_mas_wbs_ServiceLocator();
// 设置 WebService 方式通信适配插件服务地址
cmcc_mas_wbs_ServiceLocator.setcmcc_mas_wbsEndpointAddress("WebService 方式通信适配插件服务地址");
// 生成客户端访问接口
Cmcc_mas_wbs_PortType cmcc_mas_wbs_PortType =
cmcc_mas_wbs_ServiceLocator.getcmcc_mas_wbs();

// 向通信适配插件提交发送彩信的请求
SendMessageRequest sendMessageRequest = new SendMessageRequest();
// 设置消息要被发送到的地址
sendMessageRequest.setAddresses(addresses);
// 设置应用 ID 或插件 ID
sendMessageRequest.setApplicationID(applicationID);
// 设置和消息一起发送的数据
sendMessageRequest.setContent(content);
// 设置由该应用填写的应用内部扩展号码。MAS 服务器需自动补充为此业务分配的服务号码
sendMessageRequest.setExtendCode(extendCode);
// 设置消息的优先级
sendMessageRequest.setPriority(priority);
// 设置是否需要返回递交状态报告。true 表示需要，false 表示不需要。若无，则不返回
sendMessageRequest.setReceiptRequest(receiptRequest);
// 设置消息的主题
sendMessageRequest.setSubject(subject);
// 发送彩信（返回彩信发送响应）
SendMessageResponse sendMessageResponse =
cmcc_mas_wbs_PortType.sendMessage(sendMessageRequest);

```

4.5.2.2. 获取彩信发送状态-getMessageDeliveryStatus 操作

4.5.2.2.1. 操作说明

获取已提交发送的消息的发送状态。requestIdentifier 参数为发送请求标识。

4.5.2.2.2. 开发代码举例

输入：getMessageDeliveryStatusRequest

参数	类型	可选性	描述
ApplicationID	xsd:string	M	应用ID或插件ID
requestIdentifier	xsd:string	M	发送请求标识。

输出：getMessageDeliveryStatusResponse

参数	类型	可选性	描述
Result	DeliveryInformation [0..unbounded]	M	消息发送状态。

代码实现如下：

```

Cmcc_mas_wbs_ServiceLocator cmcc_mas_wbs_ServiceLocator = new
Cmcc_mas_wbs_ServiceLocator();
// 设置 Webservice 方式通信适配插件服务地址
cmcc_mas_wbs_ServiceLocator.setcmcc_mas_wbsEndpointAddress("WebService 方式通
信适配插件服务地址");
// 生成客户端访问接口
Cmcc_mas_wbs_PortType cmcc_mas_wbs_PortType =
cmcc_mas_wbs_ServiceLocator.getcmcc_mas_wbs();

// 获取已提交发送的消息的发送状态的请求
GetMessageDeliveryStatusRequest getMessageDeliveryStatusRequest = new
GetMessageDeliveryStatusRequest();
// 设置应用 ID 或插件 ID
getMessageDeliveryStatusRequest.setApplicationID(applicationID);
// 设置发送请求标识
getMessageDeliveryStatusRequest.setRequestIdentifier(requestIdentifier);
// 获取已提交发送的消息的发送状态（返回发送状态）
org.csapi.www.schema.mms.DeliveryInformation[] adeliveryInformation =
cmcc_mas_wbs_PortType.getMessageDeliveryStatus(getMessageDeliveryStatusRequest);

```

4.5.2.3. 通知彩信发送状态-notifyMessageDeliveryReceipt 操作

4.5.2.3.1. 操作说明

用于通知应用系统或应用插件已发送彩信的发送状态，由适配插件发起。

4.5.2.3.2. 开发代码举例

输入：notifyMessageDeliveryReceiptRequest

参数	类型	可选性	描述
Correlator	xsd:string	M	标识某一个消息发送请求。
deliveryStatus	DeliveryInformation [0..unbounded]	M	消息发送状态

输出：notifyMessageDeliveryReceiptResponse

4.5.2.4. 获取 M0 彩信摘要-getReceivedMessages 操作

4.5.2.4.1. 操作说明

应用及插件通过此操作接收彩信，在请求中指明优先级。若有发往该应用系统或应用插件的彩信，则在响应中列出。

4.5.2.4.2. 开发代码举例

输入：getReceivedMessagesRequest

参数	类型	可选性	描述
ApplicationID	xsd:string	M	应用ID或插件ID
priority	MessagePriority	O	指定要接收的彩信的优先级，要

			求返回高于或等于该优先级的彩信。若未指定，则返回所有彩信。
--	--	--	-------------------------------

输出：getReceivedMessagesResponse

参数	类型	可选性	描述
receivedMessage	MessageReference [0..unbounded]	O	包括一组收到根据指定过滤注册标识符和优先权的消息。

代码实现如下：

```

Cmcc_mas_wbs_ServiceLocator cmcc_mas_wbs_ServiceLocator = new
Cmcc_mas_wbs_ServiceLocator();
// 设置 Webservice 方式通信适配插件服务地址
cmcc_mas_wbs_ServiceLocator.setcmcc_mas_wbsEndpointAddress("WebService 方式通
信适配插件服务地址");
// 生成客户端访问接口
Cmcc_mas_wbs_PortType cmcc_mas_wbs_PortType =
cmcc_mas_wbs_ServiceLocator.getcmcc_mas_wbs();

// 接收彩信的请求
GetReceivedMessagesRequest getReceivedMessagesRequest = new
GetReceivedMessagesRequest();
// 设置应用 ID 或插件 ID
getReceivedMessagesRequest.setApplicationID(applicationID);
// 设置要接收的彩信的优先级，要求返回高于或等于该优先级的彩信。若未指定，
则返回所有彩信
getReceivedMessagesRequest.setPriority(priority);
// 接收彩信（返回接收到的彩信信息）
org.csapi.www.schema.mms.MessageReference[] messageReference =
cmcc_mas_wbs_PortType.getReceivedMessages(getReceivedMessagesRequest);

```

4.5.2.5. 获取 M0 彩信-getMessage 操作

4.5.2.5.1. 操作说明

读取整条彩信。在请求中指定消息的标识（该标识对应 getReceivedMessagesResponse 中 MessageReference 的 messageId 参数），在返回的响应中，包含与消息一起接收到的数据。

4.5.2.5.2. 开发代码举例

输入：getMessageRequest

参数	类型	可选性	描述
ApplicationID	xsd:string	M	应用ID或插件ID
messageRefIdentifier	xsd:string	M	消息标识

输出：getMessageResponse

参数	类型	可选性	描述
mmsMessage	xsd: MmsMessage	No	彩信消息

代码实现如下：

```

Cmcc_mas_wbs_ServiceLocator cmcc_mas_wbs_ServiceLocator = new
Cmcc_mas_wbs_ServiceLocator();
// 设置 Webservice 方式通信适配插件服务地址
cmcc_mas_wbs_ServiceLocator.setcmcc_mas_wbsEndpointAddress("WebService 方式通
信适配插件服务地址");
// 生成客户端访问接口
Cmcc_mas_wbs_PortType cmcc_mas_wbs_PortType =
cmcc_mas_wbs_ServiceLocator.getcmcc_mas_wbs();

// 接收彩信的请求
GetMessageRequest getMessageRequest = new GetMessageRequest();
// 设置应用 ID 或插件 ID
getMessageRequest.setApplicationID(applicationID);
// 设置要接收的彩信的优先级，要求返回高于或等于该优先级的彩信。若未指定，
则返回所有彩信
getMessageRequest.setMessageRefIdentifier(messageRefIdentifier);
// 接收彩信（返回接收到的彩信信息）
GetMessageResponse getMessageResponse =
cmcc_mas_wbs_PortType.getMessage(getMessageRequest);

```

4.5.2.6. 通知 MO 彩信到达-notifyMessageReception 操作

4.5.2.6.1. 操作说明

用于通知应用系统或应用插件接收到的彩信。在请求中包含彩信的所有信息及数据。

4.5.2.6.2. 开发代码举例

输入：notifyMessageReceptionRequest

参数	类型	可选性	描述
ApplicationID	xsd:string	M	应用 ID 或插件 ID
Message	MessageReference	M	与接收消息相关的所有信息
Content	xsd:string	O	与消息一起接收到的数据。

输出：notifyMessageReceptionResponse

4.5.3. Wappush 短信接口

4.5.3.1. 发送 Wappush-sendPush 操作

4.5.3.1.1. 操作说明

本操作请求把一个消息的 URL 推送到一组目标地址，并且返回一个标识 requestIdentifier 以唯一标识这个消息发送请求。应用系统或应用插件可以用 requestIdentifier 查询消息发送状态。

4.5.3.1.2. 开发代码举例

输入：sendPushRequest

参数	类型	可选性	描述
----	----	-----	----

ApplicationID	xsd:string	M	应用 ID 或插件 ID。
addresses	xsd:anyURI [1..unbounded]	M	推送的目标终端设备的地址集
targetURL	xsd:anyURI	O	推送的 URL 连接
ExtendCode	xsd:string	O	指由该应用系统或应用插件填写的扩展号码。MAS 服务器需自动补充为此业务分配的服务代码。
subject	xsd:string	O	主题（可选）
receiptRequest	xsd:boolean	O	指示是否需要网络侧返回递交状态报告。true 表示需要返回状态报告，false 表示不需要。若无，则不返回。

输出：sendPushResponse

参数	类型	可选性	说明
requestIdentifier	xsd:string	M	用来标识一次消息发送。

代码实现如下：

```

Cmcc_mas_wbs_ServiceLocator cmcc_mas_wbs_ServiceLocator = new
Cmcc_mas_wbs_ServiceLocator();
// 设置 Webservice 方式通信适配插件服务地址
cmcc_mas_wbs_ServiceLocator.setcmcc_mas_wbsEndpointAddress("WebService 方式通
信适配插件服务地址");
// 生成客户端访问接口
Cmcc_mas_wbs_PortType cmcc_mas_wbs_PortType =
cmcc_mas_wbs_ServiceLocator.getcmcc_mas_wbs();

// 推送请求
SendPushRequest sendPushRequest = new SendPushRequest();
// 设置推送的目标终端设备的地址集
sendPushRequest.setAddresses(addresses);
// 设置应用 ID 或插件 ID
sendPushRequest.setApplicationID(applicationID);
// 设置由该应用系统或应用插件填写的扩展号码。MAS 服务器需自动补充为此业务
分配的服务代码
sendPushRequest.setExtendCode(extendCode);
// 设置是否需要网络侧返回递交状态报告。true 表示需要返回状态报告，false 表示不
需要。若无，则不返回
sendPushRequest.setReceiptRequest(receiptRequest);
// 设置主题（可选）
sendPushRequest.setSubject(subject);
// 设置推送的 URL 连接
sendPushRequest.setTargetURL(targetURL);
// 发送 PUSH 消息（返回发送状态）
SendPushResponse sendPushResponse =

```



```
cmcc_mas_wbs_PortType.sendPush(sendPushRequest);
```

4.5.3.2. 获取 Wappush 发送状态-getPushDeliveryStatus 操作

4.5.3.2.1. 操作说明

应用系统或应用插件通过本操作查询先前提交的 WAP PUSH 请求的发送状态。

4.5.3.2.2. 开发代码举例

输入：getPushDeliveryStatusRequest

参数	类型	可选性	说明
ApplicationID	xsd:string	M	应用 ID 或插件 ID。
requestIdentifier	xsd:string	M	标识一次消息推送请求。

输出：getPushDeliveryStatusResponse

参数	类型	可选性	说明
result	DeliveryInformation [1 .. unbounded]	M	推送状态

代码实现如下：

```
Cmcc_mas_wbs_ServiceLocator cmcc_mas_wbs_ServiceLocator = new
Cmcc_mas_wbs_ServiceLocator();
// 设置 Webservice 方式通信适配插件服务地址
cmcc_mas_wbs_ServiceLocator.setcmcc_mas_wbsEndpointAddress("WebService 方式通
信适配插件服务地址");
// 生成客户端访问接口
Cmcc_mas_wbs_PortType cmcc_mas_wbs_PortType =
cmcc_mas_wbs_ServiceLocator.getcmcc_mas_wbs();

// 获取发送状态的请求
GetPushDeliveryStatusRequest getPushDeliveryStatusRequest = new
GetPushDeliveryStatusRequest();
// 设置应用 ID 或插件 ID
getPushDeliveryStatusRequest.setApplicationID(applicationID);
// 设置请求标识
getPushDeliveryStatusRequest.setRequestIdentifier(requestIdentifier);
// 获取 PUSH 发送状态（返回发送状态集合）
org.csapi.www.schema.wap.DeliveryInformation[] deliveryInformation =
cmcc_mas_wbs_PortType.getPushDeliveryStatus(getPushDeliveryStatusRequest);
```

4.5.3.3. 通知 Wappush 发送状态-notifyPushDeliveryReceipt 操作

4.5.3.3.1. 操作说明

本操作由适配插件向应用侧通知推送结果。

4.5.3.3.2. 开发代码举例

输入：notifyPushDeliveryReceiptRequest

参数名称	类型	可选性	说明
requestIdentifier	xsd:string	M	标识一次消息推送请求。

deliveryStatus	DeliveryInformation [1 .. unbounded]	M	推送状态
----------------	---	---	------

输出: notifyPushDeliveryReceiptResponse

4.6. 常见错误及处理方式

业务异常:

- SVC0001 - 业务错误
- SVC0002 - 不合法的输入值（如接收地址为空）
- SVC0004 - 不合法地址
- SVC0006 - 不合法的组
- SVC0280 - 消息过长
- SVC0281 - 未知的数据格式
- SVC0283 - 不支持下发到达通知

策略异常:

- POL0001 - 策略错误
- POL0003 - 地址超界
- POL0006 - 组不被允许
- POL0007 - 嵌套的组不被允许
- POL0008 - 不允许支付
- POL0900 - 不支持群发
- POL0906 - 不被允许操作或者请求（如不支持 OA 业务能力短信发送、应用 ID 或插件 ID 不存在等）
- POL0907 - 不允许群发
- POL0908 - 短信消息超长
- POL0910 - 通信未建立（如短信网关连接不正常、彩信网关连接不正常）
- POL9001 - 超过群发提交的数量限制
- POL9002 - 禁止时间段不允许提交短信

5. 接口描述

5.1. 短信接口

5.1.1. 接口说明

集团客户应用系统和应用插件通过此接口收发短消息。WebService 方式通信适配插件收到应用插件的发送请求后，通过 MAS 服务器基座中的 SMS 通信协议模块发送 SMS；当从 SMS 通信协议模块收到 SMS 后，通过此接口将短消息发送给集团客户应用系统和应用插件。

5.1.2. 流程说明

短消息接口包括发送和接收两个流程。



图 5-2 短消息发送流程示意图

发送短信由应用系统或应用插件发起，通过 `sendSms` 操作实现。在提交短信发送请求之后，应用侧可以主动获取短信发送状态（`getSmsDeliveryStatus` 操作），或由 Webservice 方式通信适配插件通知应用侧短信发送状态（`notifySmsDeliveryStatus` 操作）。

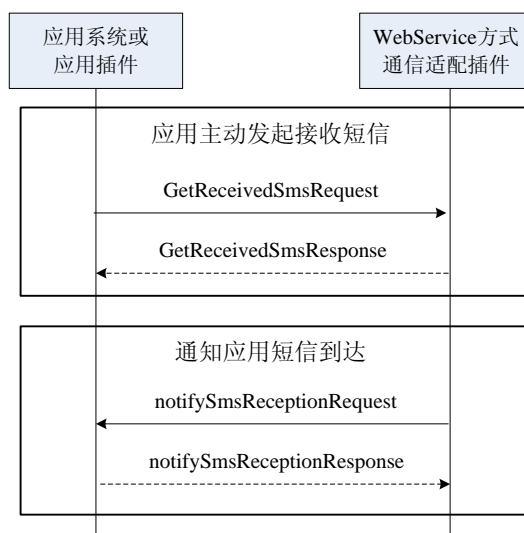


图 5-3 短消息接收流程示意图

接收短信可以通过两种方式，一种是由应用侧发起（GetReceivedSms 操作），WebService 方式通信适配插件将所接收到的发往该应用的短信在响应中提交；第二种是由 Webservice 方式通信适配插件通知应用侧有短信到达（notifySmsReception 操作），短信内容包含在请求中。

5.1.3. 数据类型

DeliveryStatus

含义：传送状态

数据类型：枚举，定义如下：

参数	描述
Delivered	短消息已成功递交。
DeliveryUncertain	递交状态未知：例如，因为短消息被发送到另外一个网络。
DeliveryImpossible	无法成功发送；短消息在超时前无法被递交。
MessageWaiting	消息仍在排队等待递交。这是一个临时状态，等待转换为前述的状态之一。
DeliveryToTerminal	短消息已发给终端。
DeliveryNotificationNotSupported	不支持短消息提交通知。
KeywordFilterFailed	关键字过滤未通过。

DeliveryInformation

含义：传送消息

数据类型：结构。定义如下：

参数	类型	描述
Address	xsd:anyURI	目的地址信息。
DeliveryStatus	DeliveryStatus	发送状态。

SMSMessage

含义：短消息信息

数据类型：结构。定义如下：

参数	类型	描述
Message	xsd:string	短消息中的文本。
SmsServiceActivationNumber	xsd:anyURI	与被调用的消息业务相关的号码，即，终端用来发送消息的目标地址。
SenderAddress	xsd:anyURI	指示短消息发送者的名称，即作为消息发起者显示在用户终端上的名称。
MessageFormat	MessageFormat	编码格式。

MessageFormat

含义：消息编码类型

数据类型：枚举。定义如下：

参数	描述
ASCII	ASCII 字符。
UCS2	USC2 格式的 UniCode 字符。
GB18030	GB18030 格式的中文字符。
GB2312	GB2312 格式的中文字符。
Binary	二进制短信，用十六进制字符串。

SendMethodType

含义：发送选项

数据类型：枚举。定义如下：

参数	描述
Normal	普通短信
Instant	普通短信立即显示
Long	长短信
Structured	长度小于 160 字节，但 UDHI 需置为 1

5.2. 彩信接口

5.2.1. 接口说明

应用系统或应用插件调用发送接口给目的用户发送一条彩信。通信适配插件采用非同步的通知机制将彩信提交状态通知给应用系统或应用插件。对于彩信接收，可以由通信适配插件主动通知应用系统或应用插件有彩信到达，应用系统或应用插件也可通过接口从通信适配插件接收彩信。

5.2.2. 流程说明



图 5-4 发送彩信流程示意图

发送彩信由应用系统或应用插件发起，通过 `sendMessage` 操作实现。在彩信发送之后，应用侧可以主动获取彩信发送状态（`getMessageDeliveryStatus` 操作），或由 Webservice 方式

通信适配插件通知应用侧彩信发送状态（notifyMessageDeliveryReceipt 操作）。

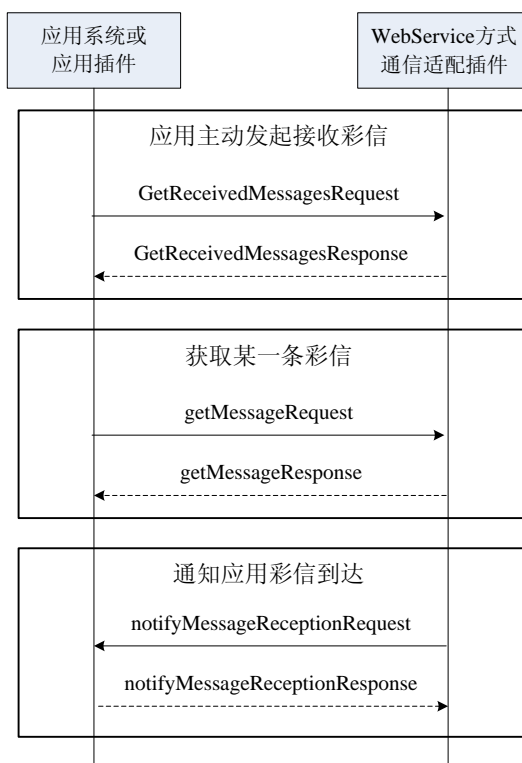


图 5-5 接收彩信流程示意图

接收短信可以通过两种方式，一种是由应用侧发起（GetReceivedMessages 操作），WebService 方式通信适配插件将发往该应用的彩信的标识在响应中提交，应用侧获取标识之后，可以选择获取某一条彩信（getMessage 操作）；第二种是由 Webservice 方式通信适配插件通知应用侧有彩信到达（notifyMessageReception 操作），在请求中包括彩信内容。

5.2.3. 数据类型

DeliveryStatus

数据类型：枚举

名称	描述
DeliveredToNetwork	消息成功发送至网络
DeliveryUncertain	消息下发状态不确定
DeliveryImpossible	消息无法发送。
MessageWaiting	消息在队列中，尚未发送。
DeliveredToTerminal	消息成功发送至终端。
DeliveryNotificationNotSupported	不支持消息下发收到通知

MessagePriority

数据类型：枚举

名称	描述
----	----

Default	缺省消息优先级
Low	低的消息优先级
Normal	正常的消息优先级
High	高的消息优先级

DeliveryInformation

数据类型：结构

参数	类型	描述
address	xsd:anyURI	消息接收地址
deliveryStatus	DeliveryStatus	发送状态

MessageReference

数据类型：结构

参数	类型	可选性	描述
messageIdentifier	xsd:string	O	指向某条彩信。如接收彩信为纯文本则此参数无效。
messageServiceActivationNumber	xsd:string	M	目的地址。
senderAddress	xsd:anyURI	M	发送方地址。
Subject	xsd:string	O	彩信的标题。
Priority	MessagePriority	M	优先级，缺省值为Normal。
Message	xsd:string	O	若接收彩信为纯文本，则此参数为消息体内容，当此参数存在时，messageIdentifier 参数无效。
dateTime	xsd:dateTime	O	消息接收时间

MmsMessage

数据类型：结构

参数	类型	可选性	描述
bodyText	xsd:string	M	彩信消息体的文本部分
Content	xsd:string	O	与消息一起接收到的数据。 MIME组包。

5.3. Wappush 短信接口

5.3.1. 接口说明

应用系统或应用插件通过此接口发送 WAP PUSH 并获取 PUSH 发送状态。

5.3.2. 流程说明

当有 WAP PUSH 消息要发送时，应用侧调用 SendPush 操作发送 PUSH 消息，之后通过 getPushDeliveryStatus 主动获取 PUSH 的发送状态。当应用系统或应用插件提供 web 服务端时，webservice 方式通信适配插件也可通过 notifyPushDeliveryReceipt 操作通知应用侧 PUSH

发送状态。

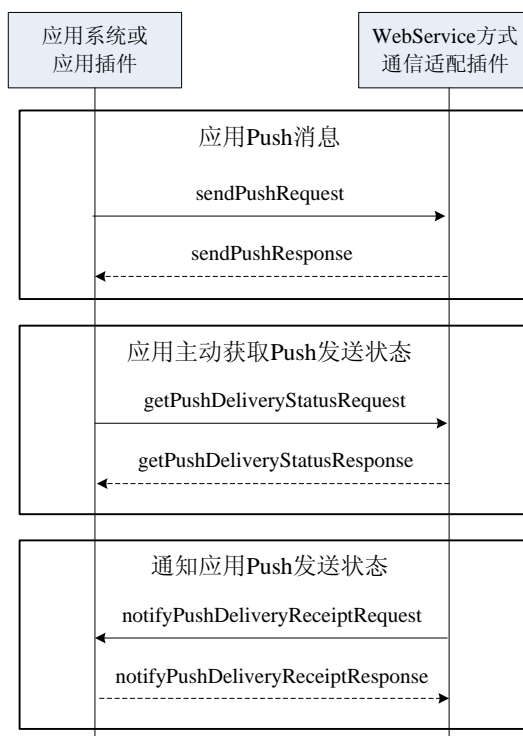


图 5-10 发送 WAP PUSH 流程示意图

5.3.3. 数据类型

DeliveryStatus

数据类型：枚举

枚举项	描述
DeliveredToNetwork	成功推送到网络。
DeliveryUncertain	推送状态未知。
DeliveryImpossible	推送不成功，比如消息超时，被客户端丢弃，或被网关拒绝。
MessageWaiting	消息正在等候被推送。
DeliveredToTerminal	成功推送到终端。
DeliveryNotificationNotSupported	不支持推送状态通知。

DeliveryInformation

数据类型：结构

结构成员	类型	描述
address	xsd:anyURI	推送请求的目的终端地址
status	DeliveryStatus	消息状态

6. 参考文档

<< MAS2.0 安装手册.doc>>

<< MAS2.0 二次开发手册(总则).doc >>