



OTA Manager V5.1

Administration Guide

All information herein is either public information or is the property of and owned solely by Gemalto NV. and/or its subsidiaries who shall have and keep the sole right to file patent applications or any other kind of intellectual property protection in connection with such information.

Nothing herein shall be construed as implying or granting to you any rights, by license, grant or otherwise, under any intellectual and/or industrial property rights of or concerning any of Gemalto's information.

This document can be used for informational, non-commercial, internal and personal use only provided that:

- The copyright notice below, the confidentiality and proprietary legend and this full warning notice appear in all copies.
- This document shall not be posted on any network computer or broadcast in any media and no modification of any part of this document shall be made.

Use for any other purpose is expressly prohibited and may result in severe civil and criminal liabilities.

The information contained in this document is provided "AS IS" without any warranty of any kind. Unless otherwise expressly agreed in writing, Gemalto makes no warranty as to the value or accuracy of information contained herein.

The document could include technical inaccuracies or typographical errors. Changes are periodically added to the information herein. Furthermore, Gemalto reserves the right to make any change or improvement in the specifications data, information, and the like described herein, at any time.

Gemalto hereby disclaims all warranties and conditions with regard to the information contained herein, including all implied warranties of merchantability, fitness for a particular purpose, title and non-infringement. In no event shall Gemalto be liable, whether in contract, tort or otherwise, for any indirect, special or consequential damages or any damages whatsoever including but not limited to damages resulting from loss of use, data, profits, revenues, or customers, arising out of or in connection with the use or performance of information contained in this document.

Gemalto does not and shall not warrant that this product will be resistant to all possible attacks and shall not incur, and disclaims, any liability in this respect. Even if each product is compliant with current security standards in force on the date of their design, security mechanisms' resistance necessarily evolves according to the state of the art in security and notably under the emergence of new attacks. Under no circumstances, shall Gemalto be held liable for any third party actions and in particular in case of any successful attack against systems or equipment incorporating Gemalto products. Gemalto disclaims any liability with respect to security for direct, indirect, incidental or consequential damages that result from any use of its products. It is further stressed that independent testing and verification by the person using the product is particularly encouraged, especially in any application in which defective, incorrect or insecure functioning could result in damage to persons or property, denial of service or loss of privacy.

© Copyright 2009 Gemalto N.V. All rights reserved. Gemalto, the Gemalto logo, GemXplore and Xxpress Campaign Technology™ (XCT™) are trademarks and service marks of Gemalto N.V. and/or its subsidiaries and are registered in certain countries. All other trademarks and service marks, whether registered or not in specific countries, are the property of their respective owners.

GEMALTO, B.P. 100, 13881 GEMENOS CEDEX, FRANCE.

Tel: +33 (0)4.42.36.50.00 Fax: +33 (0)4.42.36.50.90

Printed in France.

Document Reference: DOC117199F

Product version: 5.1.1

January 6, 2010

Contents

Preface	xvii
Additional Product Modules	xvii
For More Information	xvii
Contact Our Hotline	xviii
Chapter 1 Getting Started	1
Overview	1
Managing Cards in the Card Manager Database	2
Providing Standard Services	2
Chapter 2 What's New in OTA Manager V5.1	3
Audit Trail	3
GemConnect Scripts	3
PART I PLATFORM MANAGEMENT	
Chapter 3 Managing the Platform	7
Managing User Profiles and Accounts	7
Setting Up User Profiles	8
Setting Up User Accounts	9
Setting Up Subscriber Accounts	9
Configuring the Core Framework	11
Configuring the Invocation Registry Facility	11
Configuring the Logging Facility	12
Configuring Storage Mode	12
Storage Frequency	13
Log File Naming Convention	13
Log File Format	14
Configuring Host Mode	14
Configuring the Audit Trail	14
Configuring Billing	15
Configuring SNMP	15
Overview	15
Viewing Which SNMP Alarms Are Active	17
Configuring the Error Facility	17
Backup Configuration	17
Backing Up Data	19
Purging Data	19
System Monitoring Tools	19
Available Scripts	19
Using the Scripts	20
FileSystemCheck.sh	20
HardwareMonitoring.sh	20
Statistical Tools	22
Available Scripts	22
Using the Scripts	22

Chapter 4	Managing Products	25
Managing a Product's Life Cycle	25	
Using the Management Interface	25	
Using the gemconnect Script	26	
The Product Life Cycle	28	
Starting a Product	28	
Stopping a Product	28	
Killing a Product	28	
Managing Multiple Card Manager Instances	28	
Configuring a Card Manager Instance	29	
Editing the Trace Level	30	
Modifying Product Parameters	31	
Activating and Deactivating Billing	32	
Activating and Deactivating the Audit Trail	32	
Managing Channel Drivers	33	
Setting the SMS Channel Driver Velocity	34	
Managing Formatting Libraries	34	
How the Platform Selects a Formatting Library	36	

PART II USER TASKS

Chapter 5	Managing Services	39
Introduction	39	
Adding Services	40	
Batchloading Service Declarations and Implementations	40	
Associating Service Implementations with a Card Profile	42	
Chapter 6	Provisioning Card Profiles	43
Introduction	43	
Managing Card Profiles	45	
Defining Card Vendors	46	
Creating Card Definitions	46	
Batchloading Card Definitions Separately	46	
Referencing a Card Definition in a Card Profile	47	
Defining Card Definitions with the Management Interface	47	
Associating Service Implementations with a Card Profile	48	
Defining Security Properties	49	
Batchloading Entity Definitions	49	
Batchloading Security Settings	51	
GSM 03.48 Security Settings	51	
ESMSV1, ESMSV2 and Native Security Settings	52	
Updating Security Settings with the Management Interface	53	
Displaying the Application List	53	
Adding Security Settings	54	
Defining the Card's Structure	56	
Specifying the Contents of Mapped Files	56	
Batchloading Card Structure Elements	56	
Defining the File System with the Management Interface	58	
Defining Application Structure with the Management Interface	59	
Deleting Elements from the Card Structure	60	
Defining Security Domain Structure with the Management Interface	61	
Defining Card Profile Properties	62	
Defining GlobalPlatform Formatting Properties	62	

Defining 03.48 Formatting Properties	63
Defining 03.40 Transport Properties	64
Chapter 7 Managing Card Instances	67
Introduction	67
Creating and Managing Groups	68
Batchloading Group Definitions	69
Creating Group Definitions in the Management Interface	70
Managing Card Instances	70
Batchloading Card Instances	72
Viewing Card Contents	73
Managing Card Security	76
The Card Security Database	76
Transport Keys	77
Creating Transport Keys	78
Specifying Transport Keys in Card Security Files	79
Batchloading Card Security Settings	81
XML File Format for Native Cards	81
XML File Format for Java Cards (Single Component Keys)	83
XML File Format for Java Cards (Multiple-Component Keys)	84
Security Mechanisms	85
Additional Security Mechanisms	85
Batchloading Card Security Files	86
Viewing Security Settings	87
Chapter 8 Submitting Service Requests	91
Submitting a Service Request	91
Using the WIR	93
Monitoring Requests	94
Request Life Cycle	94
Searching For Requests	95
Defining Queries	96
The Request Management Window	98
Suppressing a Request	99
Checking the Details of a Request	99
Chapter 9 Managing Campaigns	101
Choosing a Campaign Type	101
Setting Up and Running a Campaign	102
How the Campaign Manager Works	103
Tuning XCT Campaigns	104
Overview	104
The Pre-Processing Phase	105
The Sending Phase	105
The Post-Processing Phase	105
Step 1: Scheduling XCT Campaigns	106
Step 2: Defining Campaign Parameters	107
Setting the Validity Period	107
Setting the Grace Period	108
Setting the Maximum Number of Retries and the Retry Delay	108
Step 3: Determining SMSC Channel Parameters	108
Configuring the SMSC Channel for XCT Using Product Parameters.	108
Configuring the STACK_SIZE Product Parameter	109

Configuring SS7_MODE, INQUIRY_ACTIVATED and FLOW_CONTROL Parameters for SMSC Mode	109
Configuring SS7_MODE, INQUIRY_ACTIVATED and FLOW_CONTROL Parameters for SS7 Mode	110
Step 4: Configuring the SMSC	110
Step 5: Configuring Originating and Destination Addresses	111
If Processing PoRs	111
If Not Processing PoRs	111
Calculating the XCT Memory Footprint	112
Calculating the Invocation Cache Memory Footprint	112
Calculating the SMS Cache Memory Footprint	112
Calculating the Global Memory Footprint	113
Monitoring XCT Campaigns	113
Campaign Manager Management Interface	113
SNMP Traps and Counters	113
MRTG Statistics	114
Installing MRTG	114
MRTG Examples	114
Tuning Classic Campaigns	115
Parameters Defined at Integration Time	116
Step 1: Determining the C_COMP_ARGS Value	116
Step 2: Determining the Memory Size (MEMORY_SIZE)	116
Step 3: Determining the Size of the SMSC Buffer (SWAP_SIZE)	117
Parameters Defined by the User	118
Step 4: Determining MIN_IN_MEMORY_TIME	118
Step 5: Choosing the SMS Transmission Mode	118
Step 6: Determining the Maximum Congestion (Max Congestion)	118
Step 7: Determining the Maximum Throughput (Max Throughput)	120
Step 8: Configuring the Validity Period	120
Step 9: Determining the Maximum Number of Retries	121
Step 10: Determining the Retry Delay	121
Updating Campaign Parameters	121
Updating Campaign Manager Module Product Parameters	121
Updating a Campaign's Parameters	122
Monitoring the Current Throughput and Congestion	122
Monitoring the Current Throughput	122
Monitoring Current Congestion	123
Monitoring the Success Rate and End of Campaigns	123

PART III REFERENCE INFORMATION

Appendix A Batchloading XML Files	127
Order of Batchloading	127
The Batchloading Process	128
Card Vendor Files	128
DTD Structure	128
Example	129
Management Interface Equivalent	129
Card Definition Files	129
DTD Structure	129
Example	130
Management Interface Equivalent	131
Card Profile Files	131
DTD Structure	132
Example	137

Management Interface Equivalent	146
Service Declarations and Implementations	154
DTD Structure	155
Example	156
Management Interface Equivalent	156
Card Instances	157
DTD Structure	158
Example	158
Card Security	160
DTD Structure	164
Examples	166
Group Files	167
Example	167
Management Interface Equivalent	167
Subscriber Files	168
Example	168
Management Interface Equivalent	169
Classic Campaign Target Files	170
Example	170
XCT Campaign Target Files	170
Example	170
Appendix B Product Parameters	171
Changing Product Parameter Values	171
Exporting and Importing Product Parameters	172
Card Manager (RCA) Parameters	173
XCT Parameters	185
Appendix C Configuring SMSC Channel Drivers	187
Format of the Initialization String (InitString)	187
Configuring Channel Drivers	189
Configuring “handle_temporary_status=true” in drivers.ini	194
Configuring EXPIRED_CODE_LIST in drivers.ini	194
Configuring RETRY_SMS_ON_NACK_RESPONSE in drivers.ini	194
Configuring MSISDN_TX_CONVERT_FORMAT and MSISDN_RX_CONVERT_FORMAT in drivers.ini	195
Example 1	196
Example 2	196
Example 3	196
Example 4	196
Flow Control between OTA Manager and the SMSC	197
Pre-configured drivers.ini for Integration with BMD SMS Router	197
Determining Channel Driver Parameters (drivers.ini)	198
Supported SMSC Versions and Protocols	198
Nokia	198
SMPP	198
CMG	198
Appendix D Log File, Audit Trail, and Billing Ticket File Formats	199
Logging	199
Format of the Log File	199
Example	200
XCT Log Files	201

Audit Trail	202
CSV Format	202
Audit Trail DTD	203
Billing Ticket Format	203
Appendix E SNMP Counters, Alarms, and Traps	207
The MIB File	207
The SNMP Agent	208
Master Files	208
Additional Files	208
DTD Format	208
SNMP Counters	209
Framework Counters	209
UserAuthenticationTable	211
componentTable	211
Card ManagerCounters	211
invocationManager Table	211
cattpModuleTable	212
serviceTable	213
channelDriverTable	214
storageAreaTable	215
cardMgrTable	215
Campaign Manager Counters	216
Target Counters	217
Traffic Flow Counters	217
SNMP Traps	218
Framework Traps	219
Card Manager Traps	220
XCT Traps	221
Local Billing Facility (File Mode)	223
Local Billing Facility (Storage Mode)	224
SNMP Counters, Alarms, and Traps	225
Remote Billing Facility	225
Local Logging Facility	226
Remote Logging Facility	227
Local Audit Trail Facility	228
Remote Audit Trail Facility	229
Invocation Registry Facility	230
Database Server Facility	231
Processor CrashObserver Facility (Internal Facility)	232
Service Traps	233
SMSM Channel Manager Service	233
Card Manager Service	234
Campaign Manager Service	237
Other Traps	238
SMSM	238
Appendix F Error Messages	239
General Error Messages	239
Base System Error Messages	241
com.gemplus.gxs.api.exception Package	241
invocationregistry Package	242
snmp Package	242
container Package	242

pckExceptions Package	243
pckTimeManagement Package	243
sequencer Package	243
threadpool Package	243
toplink Package	244
java.lang Package	244
java.io Package	244
java.sql Package	244
java.net Package	245
CCI Error Codes	245
CciException Error Codes	245
MediationException Error Codes	247
BeanException Error Codes	248
ExternalExceptions Error Codes	249
Javascript Message Labels	249
Card Manager (RCA) Error Codes	249
Campaign Manager Error Codes	254
Errors Originating From the Product Connector	254
Errors Originating From Campaigns	254
Errors Originating From Campaign Scenarios	255
General Errors	255
XCT Error Codes	256
XCT Global Errors	256
XCT Pre-Processing Errors	257
XCT Sending Errors	258
XCT Post-Processing Errors	258
Appendix G Troubleshooting	261
Startup Problems	261
Problems Sending SMS Messages	262
IN PROCESS Status	264
SENDING Status	265
WAIT POR status	266
Problems with SMS Execution	266
SMS is not delivered	267
Provisioning Problems	268
Monitoring Problems	269
Problems with the Wired Internet Reader (WIR)	270
Appendix H The “Security Domain” Security Model	275
Implementing Security	275
Overview	276
Step 1: Configuring the Service Implementation	277
Step 2: Identifying the Target Application	278
Step 3: Identifying the Security Mechanisms	278
Step 4: Extracting Security Settings	279
Security Mechanisms	280
No Security	280
The Synchronization Counter	281
The Redundancy Check	281
The Cryptographic Checksum	282
Ciphering	282
The Minimum Security Level	283
Security Domains	285

Key Sets	286
Handling the Proof of Receipt	286
Backward Compatibility of Security Models	286
Appendix I GSM 03.48 Message Structure	289
GSM 03.48 Secured Packet Structure	289
Structure of the Command Packet	289
Structure of the Response Packet	293
Additional Data Response Codes	294
Appendix J CCI Configuration Properties	299
Property List Description	299
Card Manager Property	299
cc.i.invocationmanager.name	299
Request Monitoring Property	299
cc.i.monitoring.maxRequests	299
Alphabet Encoding Property	300
cc.i.encodingAlphabet.type	300
Campaign Scheduling Property	300
cc.i.campaign.schedule.destinations.size	300
Reverse Proxy Properties	300
cc.i.reverseproxy.set	300
cc.i.reverseproxy.name	300
PoR Response Option Property	300
cc.i.0348CardSecuritySettings.PoRMgtOption.set	300
Appendix K Campaign Manager Time Zones	303
References	313
OTA Transport, Security, and Remote File Management	313
SIM File System (2G)	313
(U)SIM File System (3G)	313
Others	313
Terminology	315
Abbreviations	315
Glossary	319
Index	325

List of Figures

Figure 1 - Profile List Window	9
Figure 2 - The Subscriber Search and Users List Windows	10
Figure 3 - Assigning a Subscriber's MSISDN to a Card Record	10
Figure 4 - Creating a Subscriber Internet Account	11
Figure 5 - The Production and Storage Spaces	13
Figure 6 - Overview of SNMP	16
Figure 7 - The Product Configuration and General Information Windows	26
Figure 8 - The Product Life Cycle	28
Figure 9 - The Select Instance Window Showing Card Manager Instances	29
Figure 10 - Logging: List of Domains Window	31
Figure 11 - Product Parameters List	32
Figure 12 - Channel Monitors and Channel Drivers	33
Figure 13 - Selecting a Channel Driver in the User Profile	34
Figure 14 - Formatting Libraries	35
Figure 15 - Selecting a Formatting Library	36
Figure 16 - The Card Profile List	45
Figure 17 - Service Implementations Associated with a Card Profile	48
Figure 18 - Encoding the SPI Bytes	52
Figure 19 - Encoding the KiC and KID Bytes	52
Figure 20 - The Application List	53
Figure 21 - Applet Instance Properties	54
Figure 22 - GSM 03.48 Security Settings	55
Figure 23 - The File System Window	59
Figure 24 - The Applications Structure Window	60
Figure 25 - The Security Domain Structure Window	61
Figure 26 - The Profile Properties Window	62
Figure 27 - Global Platform Formatting Properties Window	63
Figure 28 - 03.48 Formatting Properties Window	64
Figure 29 - 03.40 Transport Properties Window	65
Figure 30 - Card Content in Real Cards and OTA Manager V5.1	68
Figure 31 - The Card Search Window	71
Figure 32 - Batchloading Card Security Settings	73
Figure 33 - The SIM Card List Window	73
Figure 34 - The Card Content Window	74
Figure 35 - Properties of a File	75
Figure 36 - Properties of an Applet Instance	75
Figure 37 - Properties of a Security Domain Component	76
Figure 38 - The Card Security Database	77
Figure 39 - The Transport Key Mechanism	78
Figure 40 - The Card Manager Interface	79
Figure 41 - The Create Transport Key Window	79
Figure 42 - Batchloading Card Security Settings	86
Figure 43 - Card Security Settings Window	87
Figure 44 - Security Data Window	88
Figure 45 - Request Destination Window	92
Figure 46 - Service Drop-down List	93
Figure 47 - Criteria Setting Window	95
Figure 48 - Request Management Window	98
Figure 49 - Request Details Window	99
Figure 50 - XCT Processing Phases	104
Figure 51 - Defining Campaign Timeslots	106
Figure 52 - XCT Parameters Definition Window	107
Figure 53 - XCT Product Parameters	109

Figure 54 - TP-OA and TP-DA Addressing	111
Figure 55 - MRTG Showing Pre- and Post-Processing Throughput	114
Figure 56 - MRTG Showing Sending Throughput and Delivery Status	114
Figure 57 - SWAP_SIZE: SMSC Buffer Reserved for OTA Traffic	117
Figure 58 - The WINDOW_SIZE Parameter	118
Figure 59 - DS Received within MIN_IN_MEMORY_TIME	120
Figure 60 - DS Not Received within MIN_IN_MEMORY_TIME	120
Figure 61 - Max Congestion	122
Figure 62 - Maximum Throughput	123
Figure 63 - The Validity Period	125
Figure 64 - Card Manager Product Parameters	129
Figure 65 - Campaign Manager Module Product Parameters	129
Figure 66 - Campaign Parameters	130
Figure 67 - The Batchloading Process	136
Figure 68 - Card Vendor XML and Management Interface	137
Figure 69 - Card Definition XML and Management Interface	139
Figure 70 - Service Implementation in the Management Interface	164
Figure 71 - Defining Group Definitions in the Management Interface	175
Figure 72 - Defining Subscriber Accounts with Web Access	177
Figure 73 - Defining Subscriber Accounts without Web Access	177
Figure 74 - The Request Monitoring Window	270
Figure 75 - The Request Parameters Window	271
Figure 76 - The OTA Manager Security Model	279
Figure 77 - Overview of the Security Domain Security Model	280
Figure 78 - Service Implementation Properties	281
Figure 79 - Security Levels in the Management Interface	282
Figure 80 - Specifying Ciphering in the SPI and KiC Bytes	287
Figure 81 - Security Level Specified in the Service Implementation	288
Figure 82 - Comparing the SPI Byte 1 and the MSL	289
Figure 83 - Security Models	291
Figure 84 - User Data Header (UDH) Structure	293
Figure 85 - Command Packet Structure	294
Figure 86 - SPI Byte 1 (the Command SPI)	294
Figure 87 - SPI Byte 2 (the Response SPI)	295
Figure 88 - The KiC Byte	295
Figure 89 - The KID Byte (Release 99 and Release 5)	295
Figure 90 - The KID Byte (Release 6, Redundancy Checksum)	296
Figure 91 - The KID Byte (Release 6, Cryptographic Checksum)	296
Figure 92 - User Data Header (UDH) Structure	297
Figure 93 - Response Packet Structure	297

List of Tables

Table 1 - Log File Storage Frequencies	13
Table 2 - System Monitoring Shell Scripts	20
Table 3 - Database Statistics Scripts	22
Table 4 - GSM 03.48 Security Mechanisms	55
Table 5 - Transport Key Parameters	79
Table 6 - Transport Key Algorithms	81
Table 7 - Formatting Library V3 Algorithms	82
Table 8 - Security Algorithms	85
Table 9 - Additional Algorithms	85
Table 10 - Service Query Parameters	96
Table 11 - Choosing a Campaign Type	102
Table 12 - InitString Parameters (Campaign Manager)	119
Table 13 - Channel Driver Parameters	119
Table 14 - Card Manager Product Parameters for Campaign Tuning	122
Table 15 - Campaign Report Status Codes	131
Table 16 - Card Vendor Parameters	137
Table 17 - Card Definition Parameters	138
Table 18 - Card Profile Parameters	139
Table 19 - Service Declaration Parameters	162
Table 20 - Service Implementation Parameters	163
Table 21 - Card Content Parameters	165
Table 22 - MBCard Element Parameters	168
Table 23 - SecurityData Element Parameters	169
Table 24 - KeyValue Element Parameters	170
Table 25 - SecurityElement Parameters	170
Table 26 - KeySet Element Parameters	171
Table 27 - Security Algorithms	171
Table 28 - Card Manager (RCA) Product Parameters	181
Table 29 - XCT Product Parameters	192
Table 30 - Format of the InitString	196
Table 31 - Optional Parameters Common for all Supported Drivers	197
Table 32 - Optional Parameters for the Nokia Channel Driver	198
Table 33 - Optional Parameters for the CMG Channel Driver	199
Table 34 - Optional Parameters for the Logica/SMPP Channel Driver	200
Table 35 - Log File Fields	207
Table 36 - XCT Log Files	209
Table 37 - Audit Trail Fields	210
Table 38 - Billing Ticket Fields	212
Table 39 - An Example OID	215
Table 40 - Framework Counters	217
Table 41 - platformStatus Counters	217
Table 42 - The UserAuthentication Table	219
Table 43 - The componentTable	219
Table 44 - The invocationManager Table	219
Table 45 - The cattpModule Table	220
Table 46 - The Service Table	221
Table 47 - The channelDriver Table	222
Table 48 - The storageArea Table	223
Table 49 - The cardMgr Table	223
Table 50 - The X15McMgr Table	224
Table 51 - Campaign Counters	224
Table 52 - Target Counters	225
Table 53 - Traffic Flow Counters	225
Table 54 - gxsFacility Traps	227

Table 55 - gxsComponent Traps	228
Table 56 - gxsLifeCycle Traps	228
Table 57 - gxsCardManager Traps	228
Table 58 - gxsChannelManager Traps	229
Table 59 - gxsSMSC Traps	229
Table 60 - gxsCampaignManager Traps	229
Table 61 - X15CmMgr Traps	229
Table 62 - Local Billing Facility	231
Table 63 - Local Billing Facility	232
Table 64 - Remote Billing Facility	233
Table 65 - Local Logging Facility	234
Table 66 - Remote Logging Facility	235
Table 67 - Local Audit Trail Facility	236
Table 68 - Remote Audit Trail Facility	237
Table 69 - Invocation Registry Facility	238
Table 70 - Database Server Facility	239
Table 71 - Database Crash Observer Facility (Internal Facility)	240
Table 72 - SMSC Channel Manager Service	241
Table 73 - Card Manager Service	242
Table 74 - Campaign Manager Service Traps	245
Table 75 - SMSC Error Traps	246
Table 76 - Components Generating Error Messages	247
Table 77 - General Error Messages	247
Table 78 - Base System Error Messages	249
Table 79 - Invocation Registry Package Errors	250
Table 80 - SNMP Package Errors	250
Table 81 - Container Package Errors	250
Table 82 - pckExceptions Package Errors	251
Table 83 - pckTimeManagement Package Errors	251
Table 84 - Sequencer Package Errors	251
Table 85 - Threadpool Package Errors	251
Table 86 - Toplink package errors	252
Table 87 - java.lang Errors	252
Table 88 - java.io Package Errors	252
Table 89 - java.sql Package Errors	252
Table 90 - java.net Package Errors	253
Table 91 - CCIException Error Codes	253
Table 92 - MediationException Error Codes	255
Table 93 - BeanException Error Codes	256
Table 94 - ExternalExceptions Error Codes	257
Table 95 - Javascript Message Labels	257
Table 96 - Card Manager (RCA) Error Codes	257
Table 97 - Product Error Messages	262
Table 98 - Campaign Error Messages	262
Table 99 - Campaign Scenario Error Messages	263
Table 100 - General Error Messages	263
Table 101 - Global XCT Errors	264
Table 102 - XCT Pre-Processing Errors	265
Table 103 - XCT Sending Errors	266
Table 104 - XCT Post-Processing Errors	266
Table 105 - GSM 03.40 Errors	275
Table 106 - Specifying User of a Synchronization Counter	285
Table 107 - Key Sets	290
Table 108 -Response Status Code Values	298
Table 109 - Application-independent Errors	299
Table 110 - Memory Management Errors	299

Table 111 - Referencing Errors	299
Table 112 - Message Format Errors	300
Table 113 - Security Management Errors	300
Table 114 - Kernel and Applicative Command Errors	301
Table 115 - Proactive Command Errors	301
Table 116 - Build Application Menu Command Errors	301
Table 117 - Card Manager-Specific Errors	301
Table 118 - Remote Management	302

Gemalto LinqUs OTA Manager V5.1 is part of LinqUs Over-The-Air Suite, a fully-integrated software solution providing you with a robust, reliable and scalable infrastructure with which to manage your (U)SIM installed base. LinqUs Over-The-Air Suite allows you to:

- Deliver new applications directly to your subscribers
- Easily adapt your service portal for diverse subscriber types
- Control your subscribers' mobile environment.

This manual is intended for OTA Manager system administrators, and describes how to perform management tasks for OTA Manager.

Additional Product Modules

The following additional modules are not delivered with the standard product, but can be ordered separately for OTA Manager V5.1:

- Xxpress Campaign Technology™ (XCT™) campaign manager
- Interoperable Remote File Management (RFM) services for 3G Java cards
- OTA over IP CAT-TP module
- Wired Internet Reader (WIR) for point of sale use

For More Information

For more information on OTA Manager V5.1, see the following documents:

Document	Description
<i>Third-Party Software Installation Guide</i>	Describes how to install third-party software components, including Oracle 10g and Weblogic.
<i>Third-Party Software Prerequisites</i>	Lists the versions and licensing requirements of all third-party software packages that are necessary to install OTA Manager.
<i>Installation Guide</i>	Describes the procedure for installing the OTA Manager platform and channel drivers.
<i>Administration Guide</i>	Describes how to perform management tasks for the OTA Manager platform.
<i>Getting Started</i>	Provides an overview of OTA Manager functions, helps you understand the product and what you can do with it, and provides step-by-step tasks for using the product.
<i>Customization Guide</i>	Describes how to program a new service and add it to OTA Manager, and how to internationalize the interface.
<i>Developing Client Applications</i>	Gives information on expanding OTA Manager functionality by developing new products.
<i>Script Programming Guide</i>	Provides detailed information on how to write scenario scripts to perform remote file management tasks using the ExecScript service of OTA Manager.

Document	Description
<i>Provisioning Guide</i>	Provides the essentials for provisioning OTA Manager.
<i>Online Help</i>	Task Help (accessible from the left-hand menu bar) explains how to perform OTA Manager tasks using the graphical user interface. Field Help provides a contextual description of the fields on each window.
<i>Master Index</i>	A compilation of all OTA Manager index entries which enables you to find information in OTA Manager manuals quickly.

For information on managing applets, refer to the LinqUs Application Repository Manager (ARM) documentation.

Contact Our Hotline

If you do not find the information you need in this manual, or if you find errors, contact the Gemalto hotline at <http://support.gemalto.com/>.

Please note the document reference number, your job function, and the name of your company. (You will find the document reference number at the bottom of the legal notice on the inside front cover.)

Getting Started

Administering OTA Manager involves performing the following tasks:

- Managing user profiles and accounts
- Defining card profiles
- Creating card instances and associating them with a card profile
- Provisioning card security settings
- Starting and configuring the Card Manager core module
- Setting up billing
- Configuring SNMP, log files, audit trails, and traces
- Configuring and adding channel drivers.

Overview

The following is a typical use case for updating a subscriber's SIM card using OTA Manager:

- 1 A customer care agent receives an update request by a subscriber.
- 2 OTA Manager retrieves card type and security details for the destination MSISDN from the Card Manager (RCA) database:
 - The card's serial number
 - Service information
 - The message formatting type
 - The formatting library
 - Card content information (the current structure of the card, stored in the Card Manager database)
 - Card security information from the Card Security database.
- 3 The Message Builder module generates one or more SMS (or CAT-TP) messages for the SIM card using the data retrieved in the previous step.
- 4 OTA Manager sends the SMS (or CAT-TP) messages to the SMSC (or GGSN), which forwards them to the subscriber's mobile equipment using the GSM (or 3G) network.
- 5 The mobile equipment checks the Destination Address and the Message Class type for the final destination of the message.

- 6 The SIM card checks all the security features to validate the originator and integrity of the message.
- 7 If the security checks are satisfied, the subscriber's SIM card is updated.
- 8 Optionally, a response ("proof of receipt") is returned to OTA Manager to confirm the success of the operation or, in the case of failure, help identify the problem.

Managing Cards in the Card Manager Database

To populate the OTA Manager card database with a set of cards, you need to create the following entities in the Card Manager database:

- Card vendors
- Card definitions
- Card profiles and structures
- Card instances

Each of these entities can be created individually using the management interface, or you can use the batchload facility which uploads XML files containing definitions for several entities. For information on performing these tasks, refer to "Chapter 6 - Provisioning Card Profiles" and "Chapter 7 - Managing Card Instances".

Providing Standard Services

To provide standard services, such as **Update ADN** and **Update FDN** involves:

- Loading a service implementation
- Declaring and activating the service
- Associating the service with a card profile
- Loading card instances for the card profile

For information on performing these tasks, refer to "Chapter 5 - Managing Services".

What's New in OTA Manager V5.1

This chapter provides an overview of the new or enhanced features of OTA Manager V5.1.

Audit Trail

Audit trail generation can now be activated separately for each LinqUs Over-The-Air Suite product.

GemConnect Scripts

OTA Manager V5.1 comes with new versions of the **gemconnect** scripts in the `$PLATFORM_HOME/bin` folder. The scripts provide finer control of product instances deployed in High Availability environments.

The new scripts allow the management of many aspects of multiple host deployments, including the ability to:

- Start, stop or obtain the current internal status of any product (such as CCI commands).
- Start, stop, or obtain the current status of a container daemon on a given Virtual IP (VIP) address. The previous `container.sh` script, which requires manual post-install configuration, is now obsolete.
- Start, stop, or obtain the current status of one or more OSAGENT processes, allowing them to operate on top of a fault-tolerant CORBA bus.
- Start all products with an automatically adjustable timeout, using progressive discovery of starting products combined with the effective status inquiries on all started products.
- Correctly stop all products within an optimized period of time.

All commands are protected by timeouts which are customizable by type, product, or both.

The scripts have also been enhanced internally to make them more robust, and include useful helper commands:

- The new `gemconnect.custom` script allows you to customize almost all aspects of the scripts, from variables to shell functions. The supplied template script must be renamed to create a customization point, so is guaranteed not to be overwritten by future installations.

- Support for multiple installed solutions working together, provided they use different CORBA ports.
- Identification of running processes in situations where the Process IDentification (PID) files have been removed by error.
- Definition of four different trace levels from the command line (error, warning, information, debug).
- Ability to obtain the remote status of a product if required.
- Ability to force a method to invoke Core API commands, bypassing the preferred method of optimized discovery of available methods.
- Use of per-address/per-protocol port filtering scans whenever possible, allowing multiple different processes to listen simultaneously on the same port, but not on the same address (currently only used in the OSAGENT wrapper script).

Refer to *OTA Manager V5.1 Getting Started* for a complete description of script options.

Platform Management

Managing the Platform

This chapter describes the following platform administration tasks:

- Setting up user accounts and profiles
- Provisioning and managing subscribers
- Considerations for setting up logging, billing, and audit trails
- Configuring SNMP
- Backup policy.

Managing User Profiles and Accounts

Note: You can manage user profiles only if access to the **Profile management** function has been specified in your user profile. This is the case with the default administrator profile delivered with OTA Manager V5.1.

Each user of OTA Manager is defined by:

- A *user profile*. A user profile is a “template” describing the operations that a particular type of user is authorized to perform. Types of user profiles include administrators, customer care agents, and subscribers.
- A *user account*. A user account controls access to the OTA Manager platform. Each user account is based on and linked to a single user profile. A user account contains authentication information and specific characteristics such as account status and expiry date.

Managing user profiles and accounts involves:

- Creating user profiles for administrators, customer care agents, and subscribers (see “Setting Up User Profiles” on page 8).
- Creating administrator and customer care agent user accounts (see *OTA Manager V5.1 Getting Started*).
- Creating subscriber accounts (see “Setting Up Subscriber Accounts” on page 9).

The following user profile types can be created:

- **Administrator.** Depending on the platform functionalities specified in an administrator’s profile, an administrator can have access to any or all of the following OTA Manager management interface functions:
 - Platform management (product configuration and administration)
 - Platform monitoring (logging, audit trail monitoring, billing)
 - User Profile management (creating and managing user profiles)

- Account management (creating and managing account types)
- Subscriber management (subscriber provisioning)
- Card management (creating and managing cards, provisioning and definition of all applets that can be downloaded to cards, viewing and editing the contents of cards, card provisioning, service management, and group management).

You can also specify which services are authorized for use, quality of service settings, and the applets that users with this profile are authorized to download.

- Request management (submission of requests for all cards defined in the database, monitoring and managing of requests)
- Campaign management (campaign scenarios, campaign scheduling, and campaign monitoring)

Note: A default administrator user profile, **administratorProfile**, is created during the installation of OTA Manager V5.1.

- **Customer Care Agent.** A customer care agent has no access to platform or card management, but does have access to:

- Platform management (monitoring platform, and audit trail monitoring)
- User management (for subscribers only)
- Card management (group management, provisioning is not allowed)
- Request monitoring (submission and monitoring of their own requests, group management, and submission of request for all cards defined in the database)
- Campaign monitoring (campaign scenarios, campaign scheduling, and monitoring of own campaigns)

Note: A default customer care agent user profile, **CCAPerfile**, is created during the installation of OTA Manager V5.1.

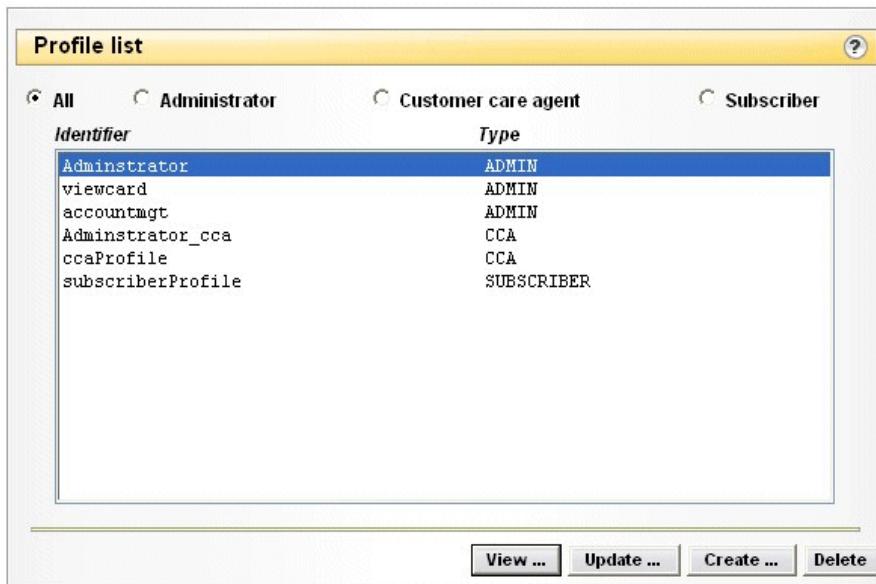
- **Subscriber.** Subscribers can access a web-based self-care interface to manage the following aspects of their accounts:

- Change account password
- View account details
- Submit requests to their own SIM card
- Monitor requests on their own SIM cards

Setting Up User Profiles

The **Profile list** window displays all the profiles in the database. On this window, you can create, update, view, or delete profiles. To display the **Profile list** window:

- 1 Do either of the following:
 - On the **Welcome** window, click **User management**.
 - Click **User** on any management interface window.
- 2 On the left-hand menu of the User management window, click **Profile**. The Profile list window is displayed with a list of all existing administrator, customer care agent, and subscriber profiles:

Figure 1 - Profile List Window

Click for details on viewing, updating, creating, or deleting user profiles.

Setting Up User Accounts

Before users can access the system, user accounts must be created and associated with the appropriate user profiles. The user profile to which a user account is associated must already exist.

OTA Manager V5.1 Getting Started describes how to create administrator and customer care agent user accounts. These accounts must exist before proceeding to set up subscriber accounts.

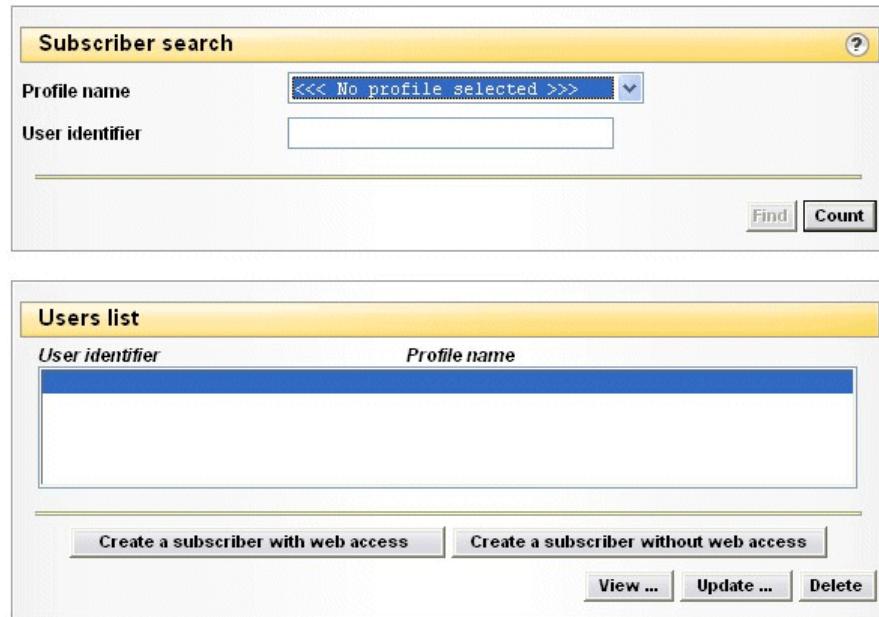
Setting Up Subscriber Accounts

Note: You can manage subscriber accounts only if access to the **Subscriber management** and **Profile management** functions have been specified in your user profile.

One of the main roles of customer care agents is to set up subscriptions for new customers. When doing this, you have the option of giving the subscriber access to an internet-based self-care interface, through which the subscriber can perform certain services, such as changing the account password or viewing account details.

To manage subscriber accounts:

- 1 Do either of the following:
 - On the **Welcome** window, click **User management**.
 - Click **User** on any management interface window.
- 2 On the left-hand menu, select **Account** and choose **Subscriber**. The **Subscriber Search** and **Users List** windows are displayed:

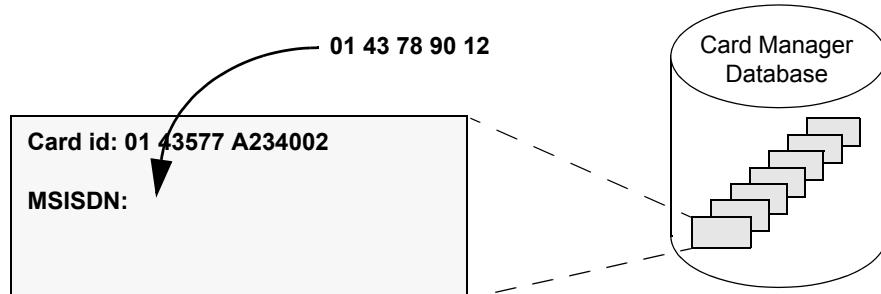
Figure 2 - The Subscriber Search and Users List Windows

- 3 Click  for details on creating new subscriber accounts (with or without web access), viewing, updating and deleting subscriber accounts using the management interface.

The customer care agent is normally responsible for providing the corresponding subscriber with a new SIM card and a phone number (the MSISDN) to use with it. The selected MSISDN is then physically stored on the SIM card, usually by means of a dedicated software program. It is important, however, to subsequently record the link between the MSISDN and the SIM card within OTA Manager V5.1.

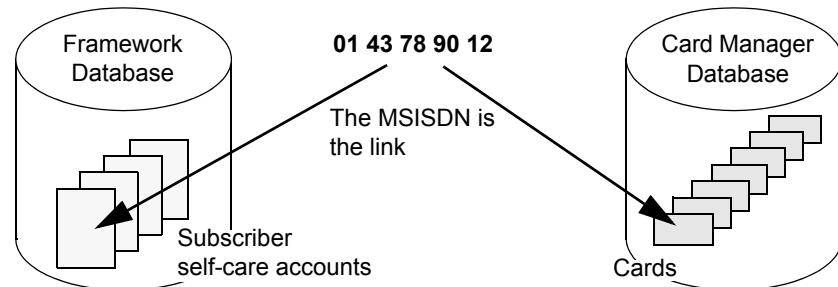
Note: If the customer care agent does not store the MSISDN on the SIM card, it is possible to store it on the card subsequently with OTA Manager V5.1, by using the **Update MSISDN** service. You *must* use the WIR channel to do this (over-the-air is not possible), and it should be done *after* following the procedure in this chapter to record the link between the MSISDN and the SIM card.

The Card Manager database in OTA Manager V5.1 already contains a record for the new SIM card, which is identified by the card serial number (the ICCID). Initially, this card record is not yet assigned to a subscriber. What you now need to do is assign the subscriber's new MSISDN to this card record, as shown below:

Figure 3 - Assigning a Subscriber's MSISDN to a Card Record

If you do this, OTA Manager V5.1 creates a record in the Framework database that contains all the subscriber's self-care account details, and links this record to the card record in the card database.

Figure 4 - Creating a Subscriber Internet Account



Note: By default, the Mobile Challenge feature generates and sends a 14-character long password. To change the length of the generated password, add the following property in the Core Framework properties file `$PLATFORM_HOME/FRWK/config/coreframework.ini`, where `$PLATFORM_HOME` is the root installation directory of the platform:

mobile.challenge.password.length=n

where `n` is the number of characters required in the password.

Configuring the Core Framework

You can configure the Core Framework by editing the file `$PLATFORM_HOME/FRWK/config/coreframework.ini`, where `$PLATFORM_HOME` is the root installation directory of the platform.

This file allows you to:

- Set database connectivity information (database URL, user login, user password and driver)
- Indicate which Management Information Base (MIB) files to consider for SNMP management.

The parameters provided by this file are considered at startup time. Changed parameter values take effect when the Core Framework is next restarted.

Configuring the Invocation Registry Facility

The Invocation Registry facility provides a repository for invocations that are managed by the Card Manager product. This facility maintains a persistent record of invocations and their execution results. Note that the Invocation Registry is only used by the Card Manager.

To configure the Invocation Registry, edit the file `$PLATFORM_HOME/FRWK/config/invocationregistry.ini`, where `$PLATFORM_HOME` is the root installation directory of the platform.

The Invocation Registry facility uses the C3P0 component to access the database. The configuration file allows you to configure any parameter supported by the C3P0 component. Each parameter is extensively commented in the file.

The parameters provided by this file are considered at start time. New parameter values take effect when the Framework is next restarted

Configuring the Logging Facility

OTA Manager V5.1 products use the logging facility to output a history of their activity. These log files are essential for debugging and support purposes.

OTA Manager V5.1 logging traces can be written to the console, stored in a file in CSV or XML format (“File” mode), stored in a file on a regular basis (“Storage” mode), or redirected to a TCP-based flow (“Host” mode).

Through the management interface you can:

- Independently activate or deactivate file and host mode storage of the log file
- Retrieve and configure the trace level for a client product
- Activate and deactivate the display of trace information on the product process console
- Display the trace levels for the root domain and sub-domains in a OTA Manager V5.1 product
- Display the current display mode of trace information.

To manage the logging facility:

- 1 Do either of the following:
 - On the **Welcome** window, click **Platform management**.
 - Click **Platform** on any management interface window.
- 2 On the left-hand menu of the **Platform management** window, click **Platform config.** and choose **Logging**.

Click  on the **Platform management** window for details.

Configuring Storage Mode

If you plan to store logs in files, you must prepare two file spaces:

- The **production space**; this space contains the current production file, where records are written.
- The **storage space**; receives the “terminated files” at a frequency that you determine. Storage space is only necessary if Storage Mode is activated.

To ensure the production space never fills, you can specify the location of the production space and the maximum amount of space allocated to it (expressed in megabytes). The Logging facility then ensures that the used space never exceeds the allocated space.

The production file is defined as a cyclic “virtual” file, made up of several smaller concrete files. The number of concrete files is arbitrarily set to 10. Only one of these files is used at any one time by the facility for record writing, which ensures that at least 90% of the logged information is available whenever the administrator looks at the contents of the files. When a file is “terminated” (that is, when its maximum size is reached, or when a specified time period has elapsed), the file is closed and another concrete file is created. Subsequent trace information is then written to this new concrete file.

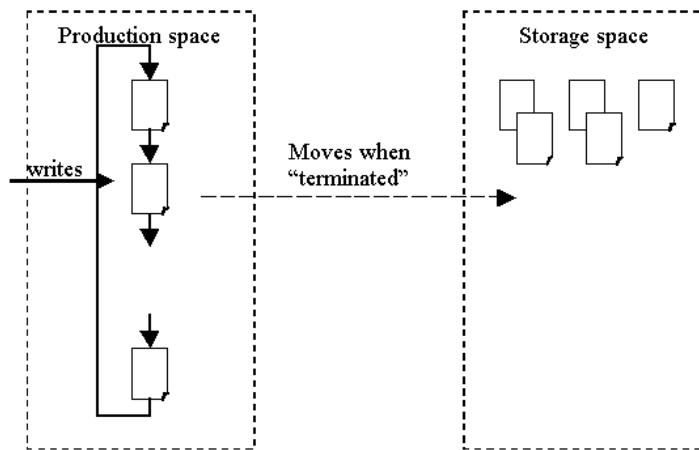
Note that if the current concrete file is the last one (that is, the tenth one), then the next concrete file returns to the first one.

You should allow a safety margin of 5% in the allocated space. As a result, each concrete file must not exceed (allocated space * 0.95)/10 MB.

If Storage Mode is deactivated, a concrete file is “terminated” only when the maximum size is reached.

If Storage Mode is activated, a “terminated” file of the production space is also automatically moved to the storage space using a particular naming convention, which ensures each file is named uniquely. A concrete file is terminated when the maximum size is reached or at the specified storage frequency (see “Storage Frequency” below).

Figure 5 - The Production and Storage Spaces



Note that this log file archiving process can be deactivated if desired, using the management interface.

Storage Frequency

Log files are written to storage at the following intervals:

Table 1 - Log File Storage Frequencies

Interval	End of period
Quarter day	12:00 midnight, 6:00 AM, 12:00 noon, 6:00 PM every day
Half day	12:00 midnight, 12:00 noon, every day
Day	12:00 midnight every day
Week	Monday 12:00 midnight every week
Custom	A logging interval, specified in milliseconds, that you determine. For example, one day is equivalent to 86400000 milliseconds.

Log File Naming Convention

When storage mode is active, terminated files are moved from the production space to storage space using the following standard naming convention:

type_ppxxxx_n.ext

where:

type = “logging”, “audit” or “billing” according to the source type.

pp = the type of interval:

QD	Quarter day
HD	Half day
DD	Day
WW	Week

xxxx = a number identifying the period, as follows:

QD	0001 to 1460 (1464 for a leap year)
HD	0001 to 0730 (0732 for a leap year)
DD	0001 to 0365 (0366 for a leap year)
WW	0001 to 0052

n = the number of the terminated file within the current period.

ext = “csv” or “xml” according to source type.

Log File Format

For details on the logging file format, refer to “Appendix D - Log File, Audit Trail, and Billing Ticket File Formats”

Configuring Host Mode

To store logging details in host mode:

- On a suitable machine in your network, create or configure a program that opens a TCP/IP socket.
- Specify the TCP/IP address and port number of the machine on the Logging Configuration window, for example, “localhost” and “8080”. Click  on the Logging Configuration window for more details.

Configuring the Audit Trail

To configure the Audit Trail facility, edit the file `$PLATFORM_HOME/FRWK/config/audittrail.ini`, where `$PLATFORM_HOME` is the root installation directory of the platform.

The Audit Trail facility uses the C3P0 component to access the database. The configuration file allows you to configure any parameter supported by the C3P0 component. Each parameter is extensively commented in the file.

The parameters provided by this file are read at systemstart time. New parameter values take effect when the Framework is next restarted.

The Audit Trail facility provides services for generating a trace of the actions performed by the users of the system.

As with logging, audit trail messages can be generated either in a file in CSV or XML format (“file mode”), stored in a file on a regular basis (“storage mode”), or redirected to a TCP-based flow (“host” mode). It is also possible to store audit trail records in OTA Manager’s Framework database (“host mode”) in order to have highest audit trail exploitation possibilities (query mechanisms, statistical analysis, and so on). You can independently activate or deactivate each of these output types.

For details on the audit trail format, refer to “Appendix D - Log File, Audit Trail, and Billing Ticket File Formats”.

To manage the Audit Trail facility:

- 1 Do either of the following:
 - On the **Welcome** window, click **Platform management**.
 - Click **Platform** on any management interface window.
- 2 On the left frame of the **Platform management** window, click **Platform config. > Audit trail**.

Click  for details.

Configuring Billing

The billing function allows you to manage the generation and storage of billing information.

Billing tickets can be generated independently for each of the OTA Manager V5.1 products. Refer to “Activating and Deactivating Billing” on page 32.

Billing ticket information is stored in Call Detail Record (CDR) format in a file on a regular basis (“storage mode”), or can be redirected to a TCP-based flow (“host” mode).

File mode manages production space and storage space as for logging (see “Configuring the Core Framework” on page 11).

For details on the billing ticket format, refer to “Appendix D - Log File, Audit Trail, and Billing Ticket File Formats”.

To manage the current platform’s billing configuration:

- 1 Do either of the following:
 - On the **Welcome** window, click **Platform management**.
 - Click **Platform** on any management interface window.
- 2 On the left frame of the **Platform management** window, click **Platform config. > Billing**.

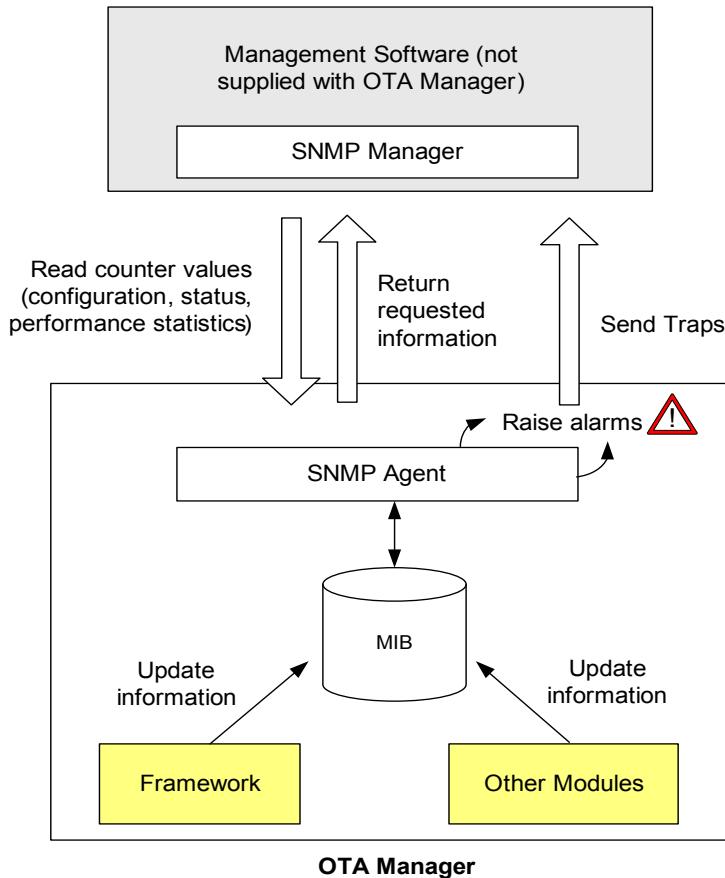
Click  for details.

Configuring SNMP

The SNMP function allows the OTA Manager V5.1 platform to manage counters and send alarms to a management platform, via the SNMP counters and trap format.

Overview

Using SNMP, a workstation running suitable network management software can remotely monitor information collected by OTA Manager V5.1’s SNMP agent. The SNMP configuration is illustrated in “Figure 6” below.

Figure 6 - Overview of SNMP

Commands and traps sent between the SNMP agent and management software are exchanged using a standard TCP/IP protocol. Configuring the SNMP connection between OTA Manager V5.1 and machines hosting network management software simply involves identifying the TCP/IP address and listening port of the SNMP manager machine.

Each alarm contains the following information:

- Code: uniquely identifies the alarm
- Timestamp
- Severity level: (critical, major, minor, warning, cancel)
- Sender Name: name of the sender
- Description: contextual message (if available)

All items of information about OTA Manager V5.1 that can be monitored using SNMP are described in a hierarchical file called a Management Information Base (MIB). In OTA Manager V5.1, additional MIB files are generated for the Framework and products. The MIB file is available for viewing in the **mibs** folder on the server.

Refer to “Appendix E - SNMP Counters, Alarms, and Traps” for details on the MIB file format and the counters and traps that are monitored by the OTA Manager V5.1’s SNMP agent.

Viewing Which SNMP Alarms Are Active

To view information on the current status of all OTA Manager V5.1 products:

- 1 Click the **Platform** button on top menu.
- 2 Click the **Supervision** button in the left-hand menu and choose the **SNMP Supervision** menu option.

The **Product list** section of the window provides information on the SNMP events for all products currently running on the platform. A green bullet indicates that no trap has been generated for the component. A red bullet indicates that a trap has been generated.

Click  for details.

Configuring the Error Facility

The Error facility provides an error repository that can be used by any product. It allows you, for example, to localize messages that are displayed by the CCI.

The Audit Trail facility uses the C3P0 component to access the database. The `errorfacility.ini` configuration file allows you to configure any parameter supported by the C3P0 component. Each parameter is extensively commented in this file.

The parameters in this file are considered at start time. Modified parameter values take effect when the Framework is next restarted.

Backup Configuration

In the following description:

- `$TOMCAT_HOME` is the root installation directory of the Tomcat web server.
- `$PLATFORM_HOME` is the root installation directory of the platform.
- `$REPORTS` is the value of the configuration parameter (GENERAL, `REPORTS_LOCATION`).
- `$LOGS_LOCATION` is the value of the “Directory location” setting for the Logging module (**Platform management > Platform config. > Logging**).
- `$LOGS_STORAGE_LOCATION` is the value of the “Storage directory name” setting for the Logging module (**Platform management > Platform Configuration > Logging**).
- `$BILLING_LOCATION` is the value of the “Directory location” setting for the Billing module (**Platform management > Platform Configuration > Billing**).
- `$BILLING_STORAGE_LOCATION` is the value of the “Storage directory name” setting for the Billing module (**Platform management > Platform Configuration > Billing**).
- `$AUDIT_LOCATION` is the value of the “Directory location” for the Audit Trail module (**Platform management > Platform Configuration > Audit Trail**).
- `$AUDIT_STORAGE_LOCATION` is the value of the “Storage directory name” setting for the Audit Trail module (**Platform management > Platform Configuration > Audit Trail**).

It is recommended to back up the following configuration data on a weekly basis.

Startup scripts:

```
$PLATFORM_HOME/FRWK/bin/*.*  
$TOMCAT_HOME/bin/*setclasspath.sh
```

Framework configuration files:

```
$PLATFORM_HOME/FRWK/config/audittrail.ini  
$PLATFORM_HOME/FRWK/config/coreframework.ini  
$PLATFORM_HOME/FRWK/config/errorfacility.ini  
$PLATFORM_HOME/FRWK/config/invocationregistry.ini
```

For each installed Card Manager instance:

```
$PLATFORM_HOME/RCA/PlugIn/services(mb/format/sms/generic0348/libv4.ini  
(plus custom dependencies defined in the file)
```

```
$PLATFORM_HOME/RCA/PlugIn/services/com/gemplus/gcota/card/  
CustomLinkFile.properties  
$PLATFORM_HOME/RCA/PlugIn/drivers/driver.ini
```

Provisioning files:

```
$PLATFORM_HOME/RCA/fileLoader/CardDefinition/*.*  
$PLATFORM_HOME/RCA/fileLoader/CardGroup/*.*  
$PLATFORM_HOME/RCA/fileLoader/CardProfile/*.*  
$PLATFORM_HOME/RCA/fileLoader/CardVendor/*.*  
$PLATFORM_HOME/RCA/fileLoader/Service/*.*  
$PLATFORM_HOME/RCA/fileLoader/ServiceImplementation/*.*  
$PLATFORM_HOME/RCA/fileLoader/SIMCard/*.*
```

Configuration files:

```
$TOMCAT_HOME/lib/resources
```

Working files:

- \$REPORTS/*.* (places where batchload reports are generated)
- \$LOGS_LOCATION/*.* (if Storage Mode is not active for the Logging module)
- \$BILLING_LOCATION/*.* (if Storage Mode is not active for the Billing module)
- \$AUDIT_LOCATION/*.* (if Storage Mode is not active for the Audit Trail module)
- **Tables** FRAMEWORK.BS_AUDIT_TRIAL, BS_AUDIT_TRAIL_ACTION,
BS_AUDIT_TRAIL_FUNCTION, BS_AUDIT_TRAIL_OBJ_TYPE (if File mode is not active
for the Audit Trail module and Database mode is active)
- \$TOMCAT_HOME/Logs/*.*

Database files:

- **Table** RCA.DRIVER_MANAGEMENT
- **Table** FRAMEWORK.INSTALL_PRD_PARAM
- **Table** FRAMEWORK.BS_COMPONENT
- **Table** FRAMEWORK.BS_COMPONENT_PARAMETER
- **Table** FRAMEWORK.BS_PRODUCT_CONFIG_DATA
- **Table** FRAMEWORK.BS_PRODUCT_TYPE
- **Table** FRAMEWORK.BS_PRODUCT_DOMAIN
- **Table** FRAMEWORK.BS_SYSTEM_CONFIG_DATA

Backing Up Data

During normal operation of the platform, the contents of databases (and in particular card-related content) are continuously being updated. It is therefore very important to back up the databases regularly. It is recommended to back up the Card Manager (RCA), Framework (FRWK), Campaign Manager and XCT databases on a nightly basis.

Purging Data

In the following, the variables are as described in “Backup Configuration” on page 17.

The following data should be purged regularly.

Provisioning files:

```
$PLATFORM_HOME/RCA/fileLoader/CardDefinition/*.*  
$PLATFORM_HOME/RCA/fileLoader/CardGroup/*.*  
$PLATFORM_HOME/RCA/fileLoader/CardProfile/*.*  
$PLATFORM_HOME/RCA/fileLoader/CardSecurity/*.*  
$PLATFORM_HOME/RCA/fileLoader/CardVendor/*.*  
$PLATFORM_HOME/RCA/fileLoader/Service/*.*  
$PLATFORM_HOME/RCA/fileLoader/ServiceImplementation/*.*  
$PLATFORM_HOME/RCA/fileLoader/SIMCard/*.*
```

where `$PLATFORM_HOME` is the root installation directory of the platform.

Working files:

- `$REPORTS/*.*` (places where batchload reports are generated)
- `$LOGS_STORAGE_LOCATION/*.*` (if Storage Mode is active)
- `$BILLING_STORAGE_LOCATION/*.*` (if Storage Mode is active)
- `$AUDIT_STORAGE_LOCATION/*.*` (if Storage Mode is active)
- `$TOMCAT_HOME/Logs/*.*`

Database files:

- Invocations table. Purging can be done using the management interface (**Platform management > Platform Configuration > Registry** and click the **Purge** button).
- Campaigns table. Purging can be done using the management interface (**Card manager > Campaign > Monitoring** and click the **Purge** button).

Note: You can purge the database only when access to the **Platform management** function has been specified in your user profile.

Note: Purging of the Audit Trail table is not necessary. The platform manages the amount of space allocated during installation.

System Monitoring Tools

A number of shell scripts are provided to help you monitor the OTA Manager platform.

Available Scripts

The scripts are located in the following directory:

```
$PLATFORM_HOME/FRWK/tools/system
```

"Table 2" list the scripts available for performing system monitoring:

Table 2 - System Monitoring Shell Scripts

Script name	Returned data
FileSystemCheck.sh	Local file systems list, with file system name and disk usage percentage
HardwareMonitoring.sh	Information about the current level of system activity

Using the Scripts

Launch the scripts from the command line using the following command:

sh script_name

FileSystemCheck.sh

This script returns a list of local system files, together with the current percentage of disk usage for each system folder. For example:

Name	Usage (%)
/	29
/boot	3
/database	63
/opt	31
/usr	43
/var	7

HardwareMonitoring.sh

Note: Under HP-UX, obtaining swap information requires privileges. Therefore, call this script using the 'root' account or it will return empty values.

This script returns details of the current levels of system activity:

- CPU:
 - **user%**: the percentage of time that the CPU is dedicated to user activity
 - **system%**: the percentage of time that the CPU is dedicated to system activity
 - **idle%**: the percentage of time that the CPU is idle
 - **iowait%**: the percentage of time that the CPU is idle waiting for disk I/O activity to complete
- RAM:
 - **total_virtual_kbytes**: total virtual memory, in kilobytes (KB)
 - **total_resident_kbytes**: total resident memory, in KB
- Swap file:
 - **total_used_kbytes**: total allocated swap file size, in KB
 - **total_free_kbytes**: total free disk space available for the swap file, in KB

The `HardwareMonitoring.sh` script has a number of command line options:

sh script_name [-c|-r|-s] [-csv] [+/-h] [-d period]

Where:

- If the **-c** option is specified, only the CPU information is displayed.
- If the **-r** option is specified, only the RAM information is displayed.
- If the **-s** option is specified, only the swap file information is displayed.

If none of the three options is specified, all fields are displayed.

- If the **-csv** option is specified, the information is returned as a header line, beginning with “#”, followed a single line with the values separated by commas. The resulting CSV file can then be easily redirected to a result file.
- If the **+h** option is passed, no header line is displayed.
- If the **-h** option is passed, the following static hardware details are returned:
 - **cpu**: The number of installed CPUs
 - **ram**: Amount of physical RAM, in KB
 - **swap**: Swap file size, in KB
- If the **-d period** option is specified, the output gives average values over the past specified period of time (default is 3 seconds).

Example 1

sh HardwareMonitoring.sh

This command displays:

```
Hardware Monitoring of the past 3 seconds
cpu : 3 0 5 92
ram : 2105384 1824976
swap : 225440 1871704
```

Example 2

sh HardwareMonitoring.sh -csv> results.txt

This command writes the following to `results.txt`:

```
# Hardware Monitoring of the past 3 seconds
cpu_user, cpu_system, cpu_idle, cpu_iowait, ram_total_virtual_kbytes,
ram_total_resident_kbytes, swap_total_used_kbytes,
swap_total_free_kbytes,
5, 1, 2, 92, 2108264, 1830540, 225440, 1871704,
```

Example 3

sh HardwareMonitoring.sh -h

This command displays:

```
cpu : 4
ram : 4031024
swap : 2097144
```

Example 4

sh HardwareMonitoring.sh -d 4

This command displays:

```
Hardware Monitoring of the past 4 seconds
cpu : 3 0 3 94
ram : 2103584 1821556
swap : 225440 1871704
```

Statistical Tools

A number of shell scripts are provided to help you gather useful database statistics for the OTA Manager platform.

Available Scripts

The scripts are provided in the following directory:

```
$PLATFORM_HOME/RCA/tools/stats/default/sqlScripts
```

The scripts are:

Table 3 - Database Statistics Scripts

Script Name	Returned Data
LUOTA_PL.sh	List of the existing card profiles, with profile identifier, profile name, and associated vendor name
LUOTA_CD.sh	The number of active, inactive or “marked for deletion” cards
LUOTA_CDM.sh	The number of active, inactive or “marked for deletion” cards, together with the associated MSISDN
LUOTA_CDWM.sh	The number of active, inactive or “marked for deletion” cards without the associated MSISDN
LUOTA_ACPP.sh	The total number of active cards for each profile
LUOTA_ACMPP.sh	The total number of active cards for each profile, together with the associated MSISDN
LUOTA_ACWMPP.sh	The total number of active cards, for each profile, without the associated MSISDN
LUOTA_ICPP.sh	The total number of inactive cards for each profile
LUOTA_ICMPP.sh	The total number of inactive cards for each profile, together with the associated MSISDN
LUOTA_ICWMPP.sh	The total number of inactive cards for each profile, without the associated MSISDN

An additional shell script, `SQL_helper_functions.sh`, is also provided. This script contains service functions called by the other scripts and cannot be run independently.

Using the Scripts

Launch the scripts from the command line as follows:

```
sh script_name USERNAME/PASSWORD@DBNAME
```

Example with `LUOTA_ACPP.sh`:

```
sh LUOTA_ACPP.sh rcaadmin/rcaadmin@rca
```

This command displays the following:

Profile_name	Total
GX_8K	999599
GX98_16K	1999600
GX3G_v2.1_64K	3000004
GXX_v3.2_128K	3000001

The scripts can also be launched with the **-c** command line option, which writes the results to a comma-separated value (CSV) file. The command to use in this case is:

sh script_name -c USERNAME/PASSWORD@DBNAME > result_file

The CSV file has the following format:

- The first line is the script title, beginning with “#”
- The second line contains column names
- The third line is blank
- The remaining lines contain the result
- Each line is separated with a carriage return
- Each column is separated with a comma (“,”)

Example with `LUOTA_ACPP.sh`:

sh LUOTA_ACPP.sh -c rcaadmin/rcaadmin@rca > results.txt

This command writes the following into `results.txt`:

```
# ACTIVE CARDS per PROFILE
Profile_name,Total,
GX_8K,999599
GX98_16K,1999600
GX3G_v2.1_64K,3000004
GXX_v3.2_128K,3000001
```

The resulting CSV file is easy to convert to, for example, an Excel table.

The scripts can also be launched in a special debug mode using the following command:

debug=true sh script_name -c USERNAME/PASSWORD@DBNAME > result_file

The debug traces are displayed on the screen and the SQL request used is stored in a temporary file in a temporary directory specified by the script. Do not forget to purge these temporary directories regularly if you use debug mode.

The debug traces are not stored in the `result_file`.

Example with `LUOTA_ACPP.sh`:

debug=true sh LUOTA_ACPP.sh -c rcaadmin/rcaadmin@rca > results.txt

This command displays the following debug traces on the screen:

```
INFO : keeping temp directory for debug
DEBUG : sqlplus rcaadmin/rcaadmin@rca : test passed
DEBUG : SqlProcessSelectQuery will run in 'CSV' mode
DEBUG : SqlProcessSelectQuery will connect to 'rcaadmin/rcaadmin@rca',
DEBUG : and execute this CSV output request :
prompt # ACTIVE CARDS per PROFILE;
prompt Profile_name,Total,;
```

```
select p.c_pro_name ||','||c.total from rca_card_profile p,
  ( select n_pro_identifier,count(*) as total from rca_smart_card where
    n_sim_state=1 group by n_pro_identifier) c where
    c.n_pro_identifier=p.n_pro_identifier;
DEBUG : sqlplus -s rcaadmin/rcaadmin@rca  @.tmp_4721/process.sql
INFO : keeping temp dir '.tmp_4721'
INFO : don't forget to remove it later...
```

The command also writes the following into results.txt:

```
# ACTIVE CARDS per PROFILE
Profile_name,Total,
GX_8K,999599
GX98_16K,1999600
GX3G_v2.1_64K,3000004
GXX_v3.2_128K,3000001
```

Managing Products

Managing a Product's Life Cycle

When managing a OTA Manager V5.1 product, you can perform any of the following actions:

- Start the product
- Shut down the product
- Kill the product

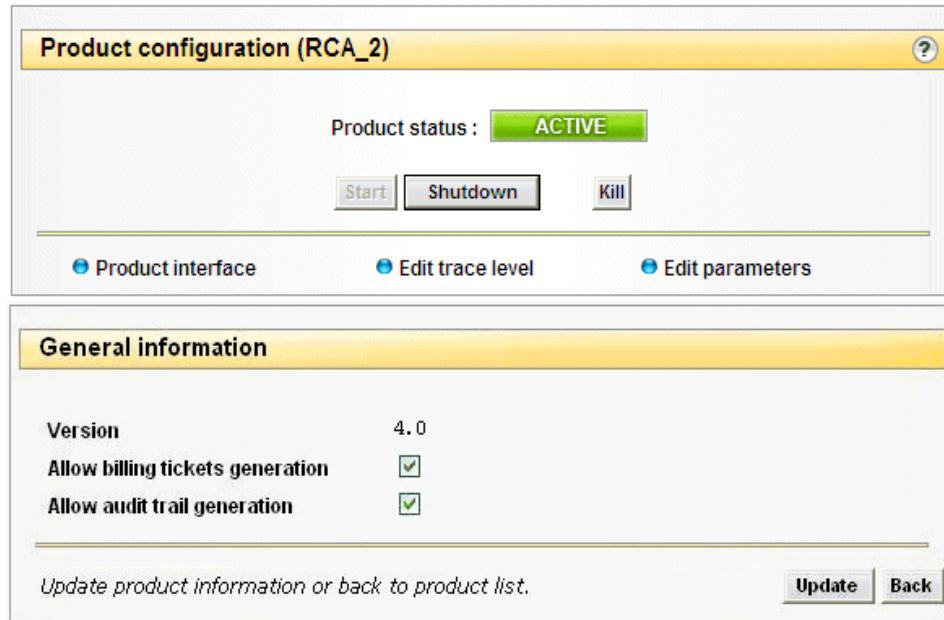
There are two possible ways to administer a product:

- 1 Using the management interface
- 2 Using the **gemconnect** script.

Using the Management Interface

To administer a product using the management interface:

- 1 Do either of the following:
 - On the OTA Manager V5.1 **Welcome** window, click **Platform management**.
 - Click **Platform** on the top menu of any OTA Manager V5.1 management interface window.
- 2 Click **Product Configuration** from the left-hand menu to display the product list.
- 3 Select a product from the list and click **Configure**. The Product Configuration and General information windows appear:

Figure 7 - The Product Configuration and General Information Windows

- 4 If the **Product status** field shows **ACTIVE**, the selected product is running normally. Otherwise, if **NOT RUNNING** is displayed, click the **Start** button to start up the product. The **NOT RUNNING** icon changes to **ACTIVE**, indicating that the product is now running.
If **MISSING** appears, the product is not installed correctly. Refer to the *OTA Manager V5.1 Installation Guide* for more information.

Using the gemconnect Script

In order to manage products' life cycle using the script interface, use the **gemconnect** command-line interface script, located in `$PLATFORM_HOME/FRWK/bin`, where `$PLATFORM_HOME` is the root installation directory of the platform.

The script supports the following commands:

- To start a product:
gemconnect -p XXX start
- To stop a product:
gemconnect -p XXX stop
- To get the status of a product:
gemconnect -p XXX status

where XXX is the name of the product instance.

For a complete description of the script's options, refer to *OTA Manager V5.1 Getting Started*.

The **gemconnect** script calls the same Core Framework methods as the management interface console described in "Using the Management Interface" on page 25, so provides all the same product management benefits with the added advantage that products can be controlled automatically (for example, when the product is used in a cluster environment with automatic failover).

All operations are associated with timeout values, whose default values are set either:

- Globally for each operation (DELAY_START_DEFAULT, DELAY_STOP_DEFAULT, and so on)
- For each product when needed (DELAY_START_PM1, and so on).

To modify the default timeout values or add new per-product values, edit the `gemconnect.custom` script. You can also override timeout values when launching the script by specifying the `-w N` option, where `N` is the timeout value in seconds.

The **gemconnect start** and **gemconnect stop** operations function on a per-contract basis :

- **start** places the product in the RUNNING state (see “Figure 8” on page 28):
 - Once the product has been started, its status is checked regularly until it is “good”. The return code is then set to OK.
 - If the OK return code is not returned within the associated timeout period, the return code is set to BAD.
 - The BAD error code indicates a fatal event that requires attention.
- **stop** places the product in the NOT RUNNING state (see “Figure 8” on page 28):
 - The **stop** command is issued. The product’s status is then checked regularly until it is “down”. The return code is then set to OK.
 - If the status check indicates that the product is still alive after the associated timeout period expires, the script kills the product using the TERM signal (`kill -15`), waits several more seconds, finalizes it with the KILL signal (`kill -9`), then returns OK.
 - The BAD error code indicates a fatal event that requires attention.
 - If a product’s PID (Process IDentification) files have been mistakenly deleted, the **stop** command cannot stop a running “unknown” product unless the **-a** option is specified to locate the actual process in the kernel process list (using a **ps** command). This causes the PID file to be regenerated.

The **gemconnect status** call combines:

- 1 A quick check of the process’s existence (using the **ps** command with the PID file)
- 2 A detailed check of the process’s status using the **getStatus** API call.

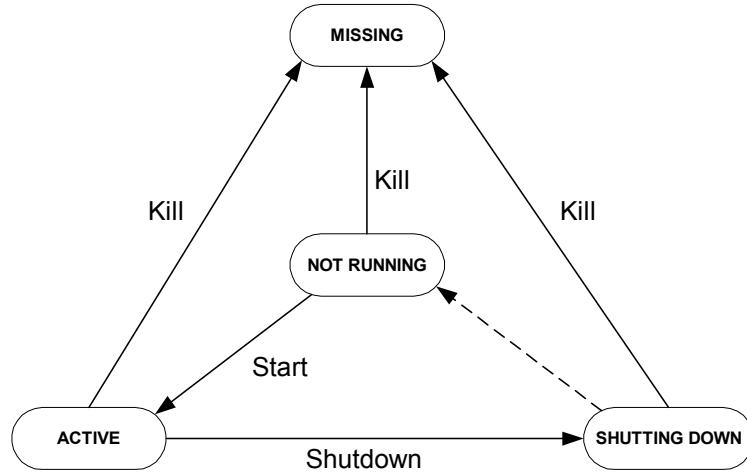
This double check ensures that the **status** call fails when launched remotely. If a product’s PID (Process IDentification) files have been mistakenly deleted, the **stop** command cannot find a running “unknown” product unless the **-a** option is specified to locate the actual process in the kernel process list (using a **ps** command). This causes the PID file to be regenerated.

Type **gemconnect -help** for a complete list of options.

The Product Life Cycle

"Figure 8" shows the product life cycle and the product status transitions that result from clicking the **Start**, **Shutdown** and **Kill** buttons:

Figure 8 - The Product Life Cycle



Starting a Product

Before starting a product, the state of the product is **NOT RUNNING**. To start the product, click **Start** on the **Product configuration** window. The status changes from **NOT RUNNING** to **ACTIVE**.

Stopping a Product

To stop a product, click **Shutdown** on the **Product configuration** window to stop the product and allow all processes to finish running. Associated processes running as part of the platform will not be stopped. The status changes from **ACTIVE** to **SHUTTING DOWN**, then to **NOT RUNNING**.

Killing a Product

This must only be performed *as an emergency action*. It is NOT recommended for normal operation. To immediately stop the product and interrupt any processes that are running, click **Kill** on the **Product configuration** window. This will interrupt any other associated processes running as part of the platform. The status changes to **MISSING**.

Managing Multiple Card Manager Instances

OTA Manager V5.1 allows you to start and manage multiple instances of the Card Manager component.

Use the Select Instance window to select an instance of the Card Manager to administer. This window appears whenever you select **Card manager** from the Welcome window or the **Products** menu and at least two instances of the standard Card Manager (RCA) or XCT-dedicated Card Manager (RCAForXCT) are active.

Figure 9 - The Select Instance Window Showing Card Manager Instances



If you are using the Card Manager for the first time, the instance selected by default is determined by the `cci.invocationmanager.name` Card Manager property (see “`cci.invocationmanager.name`” on page 299).

Otherwise, the last instance you worked with is shown in **bold** and marked with a green tick and this instance's menus are displayed on the left.

Select the instance to work with and click **Select**.

Click for further details.

Configuring a Card Manager Instance

Each Card Manager instance can be individually configured by editing the files located under:

`$PLATFORM_HOME/RCA/$INSTANCE_NAME/config/`

Where `$PLATFORM_HOME` is the root installation directory of the platform and `$INSTANCE_NAME` represents the Card Manager instance name. This can be displayed by displaying a list of installed products (click **Platform management** on the OTA Manager V5.1 Welcome window).

This directory contains the following files:

- `SCMHandler.ini`
- `c3p0.properties`
- `ehcache.xml`
- `hibernate.cfg.xml`
- `log4j.properties`
- `messagebuilder.ini`
- `treecache.xml`
- `tuning.ini`

In OTA Manager V5.1, the Card Manager uses the Hibernate open source component as a new Object Mapping layer, replacing the Toplink component used in earlier product releases. Use the `hibernate.cfg.xml` file to configure Hibernate. The most important information to configure in this file is the number of JDBC connections to use to access the Card database. In general, for optimum performance, the number of connections must be equal to the value of the `NB_THREADS` product parameter (see “`NB_THREADS`” on page 180).

Hibernate relies on the C3P0 component for database connectivity. The `c3p0.properties` file allows you to configure any parameter that is not set directly by Hibernate. This file should not require any manual modification.

Hibernate uses Log4J for internal logging. The `log4j.properties` file allows you to configure Log4J settings. This file should not require any manual modification for a production platform, but may need to be modified during the tuning phase.

Hibernate also uses the JBoss TreeCache component for second-level object caching. The `treecache.xml` file allows you to configure the behavior of this component. The most important information to configure in this file is:

- The **CacheMode** attribute. In single Card Manager instance configurations, set this attribute to “LOCAL”. In multiple Card Manager instance configurations, set this attribute to “INVALIDATION_ASYNC”.
- The **ClusterName** attribute. Only useful in multiple Card Manager instance configurations. Allows you to set the name of the Card Manager cluster. All Card Manager instances that are connected to the same database instance must have the same cluster name.

The `tuning.ini` file allows you to configure the resources allocated to a Card Manager instance. The parameters in this file are considered at start time. Changed parameter values take effect when the Framework is next restarted. Each parameter is commented extensively in the file.

The Message Builder module uses the C3P0 component to access the database. The `messagebuilder.ini` file allows you to configure any parameter supported by C3P0. Each parameter in this file is commented extensively.

The parameters provided by this file are considered at start time. New parameter values take effect when the Framework is restarted.

In addition, the Card Manager product provides numerous parameters that can be updated online. See also “Appendix B - Product Parameters” for more information.

Editing the Trace Level

Trace information in OTA Manager V5.1 is produced by each product separately. Each product has its own hierarchy of domains and sub-domains in which to group logs. Domains can be used to differentiate between debugging logs and support logs, but also to segment the overall logs flow in functional or technical modules of the product. At product initialization, a “Root” domain is automatically created (with the same name as the product) and a default trace level assigned to it.

The following trace levels are available:

- **Fatal:** This level designates severe error events that may cause the product to stop functioning.
- **Error:** This level designates errors that may still allow the product to continue running.
- **Warning:** This level designates potentially harmful situations.
- **Info:** This level designates informational messages that highlight the progress of the application at a coarse-grained level.
- **Debug:** This level designates fine-grained informational events that are most useful when debugging a product.

The goal of the trace level is to help you identify and include only the most interesting traces: for example, the **Fatal** and **Error** traces provide a good overall view of the health of the system. However, the **Fatal** and **Error** trace entries would be difficult to locate if surrounded by less important **Warning**, **Info**, and **Debug** trace entries.

It is therefore possible to set a different trace level for each individual domain. When a trace is generated in a domain, its level is compared to the parent domain's level: if the parent domain's trace level is "more verbose" than the trace level (verbosity being ranked from **Debug**, most verbose, to **Fatal**, least verbose), then the trace is generated. If the parent domain trace level is "less verbose" than the sub-domain trace level, then the trace is ignored (neither printed on screen nor written to a file).

To edit the trace level for a specific product:

- 1 On the **Product configuration** window, select the product instance, for example **RCA1 (Card Manager)** for the Card Manager.
- 2 Click **Edit trace level**. The **Logging - List of domains** window is displayed:

Figure 10 - Logging: List of Domains Window



Click for details.

Modifying Product Parameters

To view or modify product parameters:

- 1 From the **Product configuration** window (see "Figure 7" on page 26), click **Edit parameters**.
- 2 A list of product parameters appears:

Figure 11 - Product Parameters List

Product parameters selection for (RCA1)		
Selected by:	ALL	group(s)
Product parameters list		
Group	Name	Value
CATTP_MODULE	CONNECTION_INACTI...	40
CATTP_MODULE	DEVICE_HANDLER_AD...	
CATTP_MODULE	MAX_CONNECTION_DU...	600
CATTP_MODULE	MAX_CONNECTION_IN...	200
CATTP_MODULE	MAX_CONNECTION_PU...	200
CATTP_MODULE	MAX_CONNECTION_PU...	800
CATTP_MODULE	RETRANSMISSION_NB...	3
CATTP_MODULE	RETRANSMISSION_TI...	60
CATTP_MODULE	SMS_PUSH_ALPHA_ID	
CATTP_MODULE	SMS_PUSH_BEARER_DESC	
CATTP_MODULE	SMS_PUSH_BUFFER_SIZE	
CATTP_MODULE	SMS_PUSH_CATTP_DE...	10
CATTP_MODULE	SMS_PUSH_CHANNEL_ID	2
CATTP_MODULE	SMS_PUSH_NETWORK_...	
CATTP_MODULE	SMS_PUSH_OTA_IP_A...	10.10.165.49

- 3 Select the group for the parameter that you want to modify from the **Selected by** list. The list of parameters for the module displays in the **Product parameters list** window. If you select **ALL** as the group, all OTA Manager V5.1 product parameters are displayed.
- 4 Modify any of the values as necessary, then click **Update List**.

Full information on the parameter values, including default or recommended values and whether the values are “hot” or “cold” changeable, are given in “Appendix B - Product Parameters”.

You can also export the list of product parameters to an XML file, and import an XML file containing product parameters. Click  for more details.

Activating and Deactivating Billing

In the **General information** area of the Product Configuration window (see “Figure 7” on page 26), ensure **Allow billing tickets generation** is selected. This setting activates the automatic generation of billing tickets.

Caution: For billing information to be recorded, either **File mode** or **Host mode** must be activated when configuring billing. See “Configuring Billing” on page 15.

Activating and Deactivating the Audit Trail

In the **General information** area of the Product Configuration window (see “Figure 7” on page 26), ensure **Allow audit trail generation** is selected. This setting activates the automatic generation of an audit trail.

Caution: For audit trail information to be recorded, one of **File mode**, **Storage Mode**, **Host mode**, or **Database Mode** must be activated when configuring the audit trail. Refer to the online help on the Audit Trail window (**Platform > Platform Config. > Audit trail**).

Managing Channel Drivers

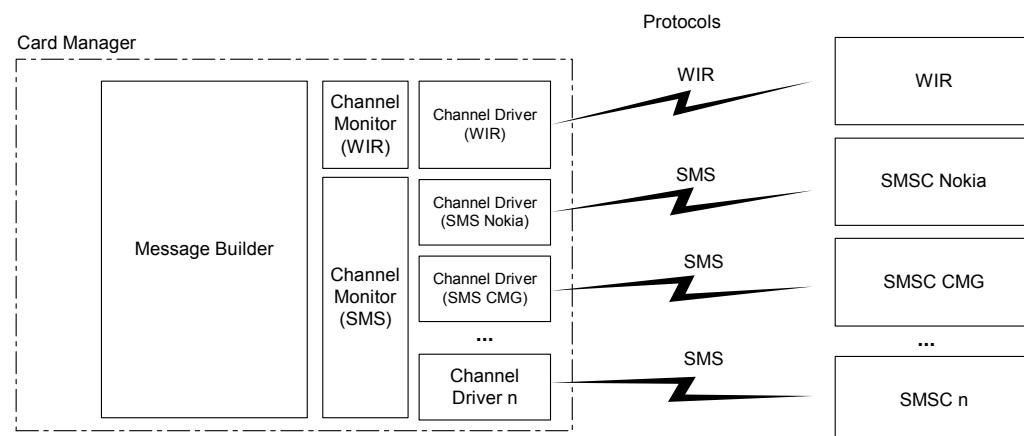
OTA Manager V5.1's Card Manager component communicates with external message systems through one or more channel monitors and various channel drivers installed during installation of the platform (see the *OTA Manager V5.1 Installation Guide*).

There are several types of channel driver available:

- SMS channel drivers allow over-the-air communications between the Card Manager on the OTA Manager platform and the subscriber, via an SMSC. Each driver allows OTA Manager V5.1 to communicate directly with the SMSC server using the SMSC manufacturer's protocol.
- The WIR channel is an optional OTA Manager V5.1 component and provides an alternative to over-the-air card modifications. The WIR channel driver allows communication between the Card Manager and a Wired Internet Reader (WIR) connected to a point of sale terminal. This channel allows a customer care agent at a point of sale to directly modify the contents or functionality of a subscriber's SIM card inserted into the WIR.
- The BIP/CAT-TP channel is an optional OTA Manager V5.1 component that provides an alternative to performing OTA card management. The BIP/CAT-TP channel driver allows communications between the Card Manager and a subscriber's SIM card over GPRS or UMTS networks using the CAT-TP protocol.

This is illustrated in "Figure 12":

Figure 12 - Channel Monitors and Channel Drivers



The **Driver Management** window allows you to manage channel drivers:

- 1 On the **Welcome** window, click **Platform**.
- 2 Click **Product config.** in the left-hand menu to display the **Product list**.
- 3 Select **Card Manager** in the list, and click **Configure**.
- 4 Click **Product interface**.
- 5 Click the **Manage** button next to **Driver Management**. This window displays the status of each channel driver and allows you to:
 - Update a channel driver's configuration, including setting the channel velocity (see "Setting the SMS Channel Driver Velocity" on page 34).
 - Delete channel drivers
 - Deactivate channel drivers
 - Activate the CRPC of a channel driver.

- Add new channel drivers.

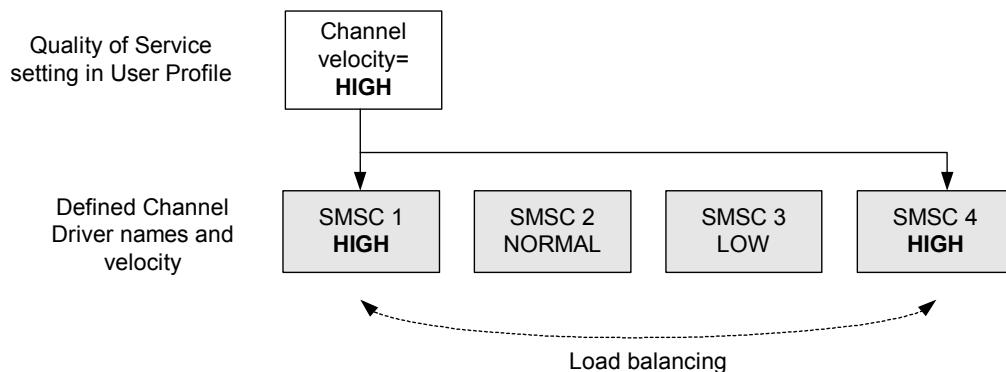
Click  for details.

OTA Manager V5.1 allows you to enable additional external connections by adding channel drivers. In this way, you can use a specialized protocol when vendor implementations are not standardized. See the *OTA Manager V5.1 Customization Guide* for instructions on how to add a new channel driver for new or updated devices.

Setting the SMS Channel Driver Velocity

When you define a service request, you can specify the channel driver with which to send SMS messages. If this optional field is not specified, the channel driver velocity specified in the user profile is used to determine the channel driver or drivers to use. If several channel drivers have the same velocity setting, the platform automatically performs load-balancing between them. This is illustrated in “Figure 13”:

Figure 13 - Selecting a Channel Driver in the User Profile



If the velocity requested in the user profile does not match any channel drivers, channel drivers with the next lowest velocity are selected instead.

The velocity setting can therefore be used to offer a higher quality of service to selected subscribers: assign a HIGH velocity in their user profile, then connect one or more channel drivers to high-speed SMSC connections.

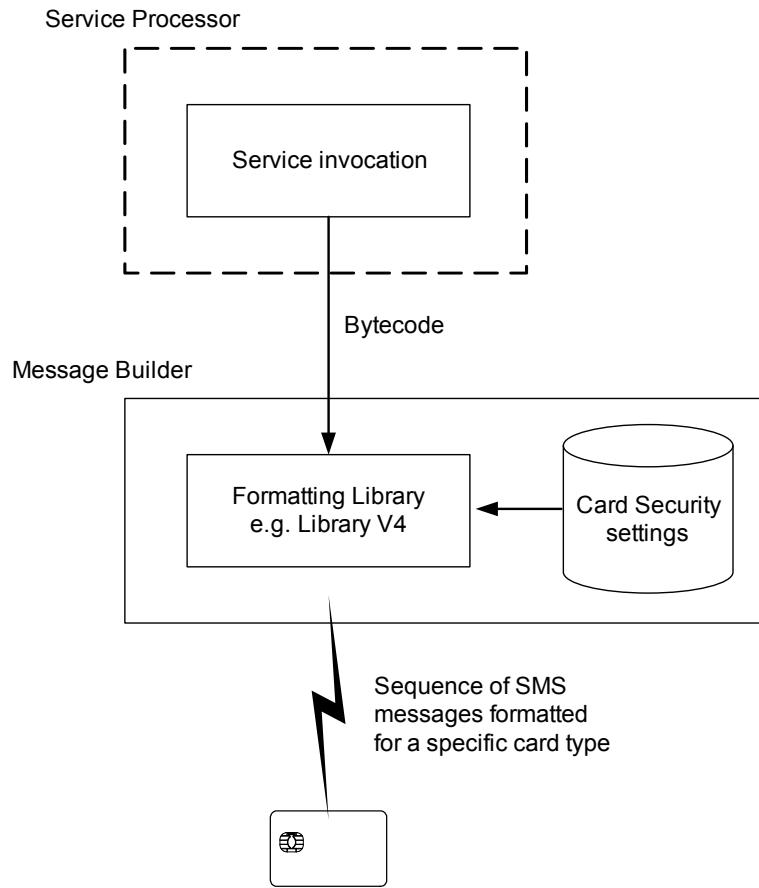
Channel driver velocity can be set to **Low**, **High**, or **Normal**. The default value is **Normal**.

Managing Formatting Libraries

OTA Manager V5.1 uses different formatting libraries to generate SMS messages for ESMS V1, ESMS V2 and GSM 03.48 transport protocols. For example, SMS messages that contain GSM 03.48 security settings destined for a Java applet on a GemXplore 3G Java Card are formatted using the Formatting Library V4, while SMS messages containing ESMS V2 security settings destined for a native application on a non-Gemalto SIM card are formatted using the Library V3 formatting library.

This is illustrated in “Figure 14”:

Figure 14 - Formatting Libraries



OTA Manager V5.1 supports a number of different formatting libraries:

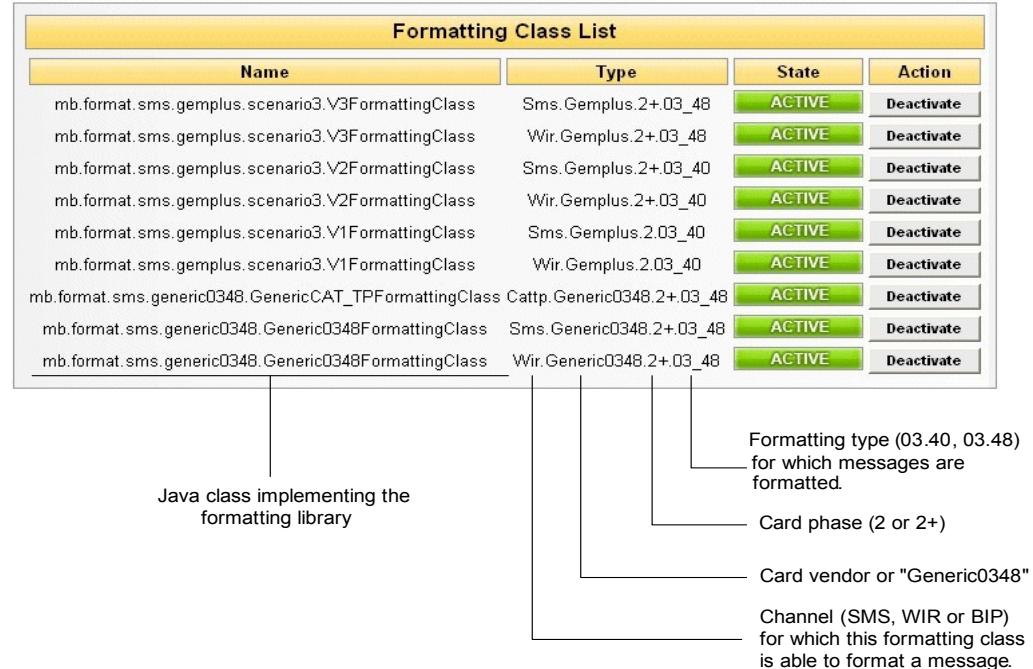
- The TP-UD Formatting Library V4. This library is compatible with 3GPP TS 23.048 version 4.2.0 Release 4, ETSI TS 102 225 Release 6, and GlobalPlatform standards. It provides support for both Gemalto and other manufacturers' interoperable cards.
- Formatting Library V3, for formatting ESMS V1, ESMS V2, and GSM 03.48 messages.
- Native formatting libraries, for formatting messages for custom non-Gemalto cards.

The TP-UD Formatting Library V4 supports a wider range of GSM 03.48 security mechanisms than the Library V3. Messages formatted by the Message Builder against the Library V4 may therefore specify security mechanisms that are not supported by the Library V3.

How the Platform Selects a Formatting Library

The following figure illustrates how OTA Manager V5.1 selects a formatting library:

Figure 15 - Selecting a Formatting Library



To display the list of available formatting libraries:

- 1 On the Platform management window, click **Product configuration**.
- 2 Select the Card Manager card product in the list, and click **Configure**.
- 3 On the Product configuration window, click **Product interface**.
- 4 Click **Formatting Class**.

Click for details on managing the formatting classes. Note that if you **Deactivate** a channel driver, all channel drivers using the same formatting class are deactivated.

Refer to “Appendix C - Configuring SMSC Channel Drivers” for a list of channel driver parameters.

User Tasks

Managing Services

This chapter describes how to provision and manage services.

Note: You can manage services only when access to the **Service management** function has been specified in your user profile. See “Managing User Profiles and Accounts” on page 7.

Introduction

OTA Manager V5.1 provides a set of value-added services that can be used to update subscribers’ SIM cards, for example, the **Update ADN** service.

Technically, each service is made up of two parts:

- 1 A *service declaration*. This is the service’s external interface. A service declaration is valid for all cards. It contains, for example, expected input data, data types, and whether the data is mandatory or optional.
- 2 A *service implementation*. A service implementation is dedicated to one particular card profile, but the same service implementation may be used for several different card profiles. A service implementation is a macro language script or Java class that generates an OTA script containing instructions to be executed by remote SIM cards.

OTA Manager V5.1 comes with a selection of predefined services to perform the most common tasks. These services are normally provisioned during installation.

You can also create and provision your own custom services—refer to the *OTA Manager V5.1 Customization Guide*.

Before services can be invoked in service requests:

- The service implementation and declaration must both be provisioned in the Card Manager’s Service repository.
- All Java class files and scripts associated with the service must be copied to the correct folders on the platform. Refer to the *OTA Manager V5.1 Customization Guide* for details.
- The service implementation must be associated with one or more card profiles.

Note that for a single service declaration, several different service implementations may be provided, each linked to a different card profile.

A service can be “profile independent”, meaning that only one implementation is provided for all card profiles. This can be true of services that require no information concerning either the card profile or the card instance to be retrieved from the Card Manager database, for example **Send Text**.

Adding Services

The procedure for adding a service to the Service repository is as follows:

- 1 Define, using the management interface or by batchloading, the service declaration and service implementation in the Card Manager database. This is described in “Batchloading Service Declarations and Implementations” that follows.
- 2 Associate the service implementations with card profiles. This is described in “Associating Service Implementations with a Card Profile” on page 42.

Batchloading Service Declarations and Implementations

To add service declarations and implementations to the platform:

- 1 Prepare an XML file defining the service declaration. Refer to “Appendix A - Batchloading XML Files” for a sample service declaration batchload file and the DTD defining the rules that the file must comply with.
- 2 The following example shows a service declaration:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE ServiceDeclModule SYSTEM "cardframework.dtd">
<ServiceDeclModule>
    <ServiceDecl name="UpdateUST_3G"
        handlerName="rca.serviceDecl.UpdateUST"
        state="1"
        isCardIndependent="false"/>
</ServiceDeclModule>
```

- 3 Prepare an XML file defining the service implementation. Refer to “Appendix A - Batchloading XML Files” for a sample service implementation batchload file and the DTD defining the rules that the file must comply with.

The following example shows a single service implementation:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE ServiceImplModule SYSTEM "cardframework.dtd">
<ServiceImplModule>
    <ServiceImpl
        name="3G_0348_UpdateUST"
        serviceName="UpdateUST_3G"
        state="1"
        javaClassName="rca.serviceImpl.generic0348.UpdateUST_3G"
        scriptName="rca/serviceImpl/generic0348/UpdateUST_3G.script"
        cardContentUpdateActivated="true"
        cardContentCheckActivated="true"
        applicationIdentifier="B00001"
        securityLevel="0" >
    </ServiceImpl>
</ServiceImplModule>
```

The attributes of the **ServiceImpl** element are described in “Table 17 - Service Implementation Parameters” on page 155. Note in particular:

- The value of the serviceName attribute must be unique and match the name of the service declaration (ServiceDecl name="UpdateUST_3G") associated with this service implementation.
- The value of the **Name** attribute must match the service implementation name that is subsequently used when associating the service implementation with a card profile (see “Associating Service Implementations with a Card Profile”).

- **cardContentUpdateActivated="true"** specifies that the content of card instances stored in the Card Manager database is to be updated upon successful completion of a service request or campaign based upon this service.
Set this option to “false” if the card instance is not to be updated. This might be the case, for example, for an **Update ADN** service request when another external application (such as Phonebook Backup) is responsible for managing card content updates.
- **cardContentCheckActivated="true"** specifies that the card instance (card content) is automatically checked in the Card Manager database.
- **applicationIdentifier**. For GSM 03.48 (3GPP TS 23.048) services, the TAR of the applet on the card that is targeted by the service to be executed. In this example, “B00001” is the TAR of the interpreter applet on GemXplore Xpresso V3 cards (different values may be used on other card types).
For Gemalto native cards, this field is a folder identifier coded on two bytes (ESMS V1), or a folder reference coded on a single byte (ESMS V2).
- **securityLevel**. The security level required by the service implementation. The security level corresponds to a set of security mechanisms that are defined in the card profile. The value can be between 0 and 99 inclusive. A value of -1 specifies that no security is to be applied. In this case, for example, the SPI bytes in a GSM 03.48 header are set to all zeros (“00 00”).

The **applicationIdentifier** and **securityLevel** parameters together define the security mechanisms that are to be applied to the OTA messages that are generated when the service is executed and transport the service data to the target card or cards. The abstraction of the security level as an integer value allows security mechanisms to be modified in card profiles without requiring modifications to the service implementations themselves.

Refer to “Chapter 8 - Submitting Service Requests” for more information on submitting service requests.

By including multiple **ServiceImpl** elements in the XML file, you can batchload multiple service implementations at the same time:

```
<ServiceImplModule>
  <ServiceImpl ....>
    ...
  </ServiceImpl>
  <ServiceImpl ....>
    ...
  </ServiceImpl>
  <ServiceImpl ....>
    ...
  </ServiceImpl>
</ServiceImplModule>
```

- 4 Batchload the service declaration and service implementation XML files from the Service List window (**Card Manager > Service** and click **Batchload**). Click  on the Service list window for further details.

The service declaration or implementation is imported to the service repository and is subsequently available in lists of available services.

Associating Service Implementations with a Card Profile

Unless a service implementation is marked as being “profile independent”, you must associate service implementations with card profiles before they can be used.

This is done when configuring the card profile. Refer to “Associating Service Implementations with a Card Profile” on page 48.

Provisioning Card Profiles

This chapter describes how to create and manage card profiles.

Introduction

A card profile is a OTA Manager internal representation of the electrical definition of a SIM card issued by the card's manufacturer. You must define card profiles for all card types that you intend managing with OTA Manager V5.1.

Once card profiles have been defined, you can proceed to associate individual card instances with the card profile—all card instances linked to a card profile then inherit the card profile's attributes. This process is described in “Chapter 7 - Managing Card Instances”.

A card profile includes:

- The **card vendor** or manufacturer of the SIM card, for example, “Gemalto”. This is described in “Defining Card Vendors” on page 46.
- The **card type** (phase 2 or 2+).
- A **card definition**. A description of the physical characteristics of the card, including:
 - The chip manufacturer
 - The card's operating system and version
 - The total amount of memory on the card, including volatile memory (RAM) and non-volatile memory (ROM and EEPROM)
 - The amount of available storage space on the card (total amount of EEPROM memory minus the size of applets already installed on the card).

This is described in “Creating Card Definitions” on page 46.

- A list of **associated service implementations** that can be used to manage the contents of the card. The service implementations associated with a card profile reflect both the card type (for example, 2G or 3G) and the remote file management (RFM) services that are to be performed on the card.

You define an initial set of associated service implementations to associate with the card profile when provisioning the card profile. The associated services can subsequently be updated using the management interface.

This is described in “Associating Service Implementations with a Card Profile” on page 48.

- Details of all **security mechanisms** implemented on the card. For example, identifiers of the security domains on the card, and whether redundancy checks or

cryptographic checksums must be implemented by OTA Manager V5.1 when sending messages to these entities on the card. Security domains are applets and applet instances for Java cards.

This is described in “Defining Security Properties” on page 49.

- The **card structure**. This includes:

- The card’s file system (MFs, DFs and EFs). You must define all files, their logical hierarchy within the file system, and their default contents (number of records, binary content of each record, and so on).
- Definitions of any applications (Java packages, applets and applet instances) that are preinstalled on the card. These might include, for example, RAM and RFM management applets and custom applets.

These files and applications may represent a complete description of the contents of the card, or they may represent only those elements that are to be managed using OTA Manager V5.1. For example, if the physical card that the card profile describes contains a custom applet that cannot be managed by OTA, this applet need not be included in the card profile.

Warning: Once the physical structure of a card profile has been declared during provisioning, you cannot subsequently delete or modify it, although new elements can be added to the structure. Refer to “Defining Application Structure with the Management Interface” on page 59.

This is described in “Defining the Card’s Structure” on page 56.

- The **Security Domain structure**. The security domain hierarchy shows extradition relationships between the security domains of the card profile. Simple applications and packages are shown in relation to their associated security domain.
- **Formatting and transport properties**. Configuration details that describe how OTA messages destined for the card are to be formatted—for example, GlobalPlatform command parameters and GSM 03.48 properties—and the transport protocol to use when downloading information to the card.

This is described in “Defining Card Profile Properties” on page 62.

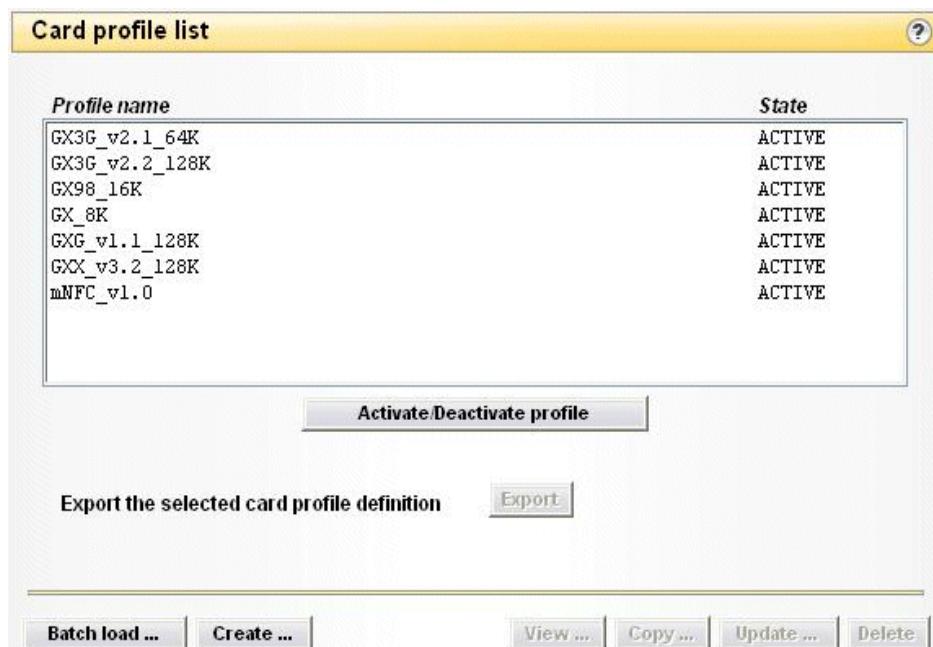
- Whether the card is compliant with formatting protocols such as GSM 03.48, ESMS V1, ESMS V2, and others.
- Whether the card profile is CAT-TP compliant.
- Whether the card profile is currently active.

Managing Card Profiles

A default card profile, for the GemXplore Xpresso V3 V2.2 card, is provisioned when installing OTA Manager V5.1. You can create copies of this profile as a basis for other card profiles, or you can create new card profiles by batchloading the information from XML files.

You can view a list of all currently provisioned card profiles on the Card Profile list window (**Card Manager > Card > Profile**):

Figure 16 - The Card Profile List



From this window, you can:

- Activate or deactivate card profiles.
- Use the batchload facility to load additional card profiles into the Card Manager database.
- Export an existing card profile to an XML file. Some elements of a card profile can only be provisioned by batchloading. To create new card profiles, therefore, you may find it easier to export a similar card profile, modify the XML file outside OTA Manager V5.1, then batchload the modified card profile. This feature is particularly useful for copying a card's structure between card profiles.
- Create, view, copy, update, and delete card profiles.

Click on the Card profile list window for detailed information on performing these tasks.

Note: The functions you can use depend on whether you have the **Card provisioning**, **Card editing**, or **Card viewing** options specified in your user profile. Buttons for functions you are not authorized to use are grayed out. See “Managing User Profiles and Accounts” on page 7.

Defining Card Vendors

The card vendor portion of the card profile provides information about card vendors or manufacturers. It is one of the items of information used by OTA Manager V5.1 to select the formatting library to use to format SMS (or CAT-TP) messages for target card types.

The Card vendor list window displays all card vendor entities in the Card Manager database.

Note: You can also batchload card vendors: refer to “Appendix A - Batchloading XML Files”.

To define card vendors:

- 1 Click **Card manager > Card > Vendor**. The **Card vendor list** window is displayed. On this window, you can:
 - Export a card vendor definition
 - Use the batchload facility to load several card vendor definitions simultaneously into the database.
 - Create, update, and delete, card vendor definitions.

Click  for details.

Creating Card Definitions

Card definitions describe the physical characteristics of a card, including the chip manufacturer, card operating system and version, and the amount of RAM, ROM and EEPROM memory on the card. You can create card definitions:

- 1 By batchloading, either:
 - Including the card definition within an XML file that contains one or more card definitions, then referencing the appropriate card definition from within the card profile. This is described in “Batchloading Card Definitions Separately” that follows.
 - Including the card definition as an integral part of the card profile definition and batchloading the card profile. This is described in “Referencing a Card Definition in a Card Profile” on page 47.
- 2 Using the management interface. This is described in “Defining Card Definitions with the Management Interface” on page 47.

Batchloading Card Definitions Separately

To batchload card definitions separately from the card profile:

- 1 Prepare an XML file defining the card definition. Refer to “Appendix A - Batchloading XML Files” for the DTD defining the rules that the file must comply with. For example:

```
<!DOCTYPE CardDefinitionModule SYSTEM "cardframework.dtd">
<CardDefinitionModule>
  <CardDefinition>
    internalName="GX3G_v2.2_128K"
    commercialName="GemXplore Xpresso V3"
    osName="GemXplore 07"
    osVersion="07.00"
```

```

    chipManufacturer="Siemens"
    chipModel="SLE66C"
    eepromSize="3200"
    ramSize="32000"
    romSize="32000"
    freeSize="64000" />
</CardDefinitionModule>
```

The **internalName** is the unique identifier of the card definition, and is used to reference the card definition from within a card profile.

The other elements refer to various software and hardware characteristics of the card. The memory size values are defined in bytes (for example, “romSize=32000” specifies that the card has 32K of ROM memory). Refer to “Figure 65 - Card Definition XML and Management Interface” on page 131 for the equivalent of the XML attributes and elements in the management interface.

- 2** Display the Card Definition list (**Card manager > Card > Definition**).
- 3** Click **Batchload**.

Referencing a Card Definition in a Card Profile

To reference a card definition from within a card profile file, specify a **cardDefinition** attribute and value. For example:

```

<CardProfileModule>
  <CardProfile name= "GX3G_v2.2_128K"
    state= "1" phase= "2+"
    cardVendor= "Gemalto"
    cardDefinition= "GX3G_v2.2_128K" ...
    ...
  </CardProfile>
</CardProfileModule>
```

The value of the **cardDefinition** attribute must *exactly* match the value of the **internalName** attribute in the card definition:

```
<CardDefinition internalName="GX3G_v2.2_128K" ...
```

Defining Card Definitions with the Management Interface

The **Card definition list** window displays all the card definition entities in the Card Manager database.

To define card definitions with the management interface:

- 1** Do either of the following:
 - On the **Welcome** window, click **Card manager**
 - Click **Card** on any other management interface window.
 - 2** On the left frame, select **Card > Definition**.
- Click  for details.

Associating Service Implementations with a Card Profile

Service implementations must be associated with one or more card profiles before they can be used in service requests or campaigns. A service implementation defines the services available and the type of message formatting required (ESMS V1, ESMS V2, or GSM 03.48).

Associating service implementations with card profiles can be achieved either by batchloading or through the management interface.

To batchload service implementation associations:

- 1 In the XML card profile file, include a list of service implementations in the **serviceImpl** attribute of the **CardProfile** element. For example:

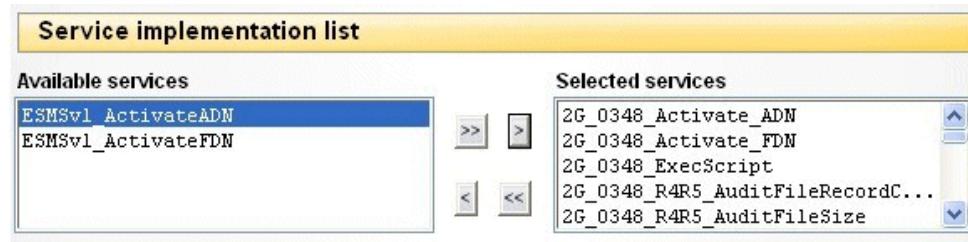
```
<CardProfile name= "GX3G" state= "1" phase= "2+"
  cardVendor= "Gemalto" cardDefinition= "GX3G_v2.2_128K"
  serviceImpl= "3G_0348_ExecScript;3G_0348_ManageHPLMNwAct;
  3G_0348_ManageOPLMNwAcT;3G_0348_ManagePLMNwAct;
  3G_0348_UpdateACC;3G_0348_UpdateADN;3G_0348_UpdateBDN;
  3G_0348_UpdateEST;3G_0348_UpdateFDN;3G_0348_UpdateFPLMN;
  3G_0348_UpdateHPLMN;3G_0348_UpdateIMSI;3G_0348_UpdateMSISDN;
  3G_0348_UpdatePL;3G_0348_UpdateSDN;3G_0348_UpdateSMSP;
  3G_0348_UpdateSPN;3G_0348_UpdateUST" >
```

The name of each service implementation must *exactly* match the service implementation name as specified in the **name** attribute of the XML file used to provision the service implementation. See “Chapter 5 - Managing Services”.

To use the management interface to associate service implementations with card profiles:

- 1 On the Card Profile List window (**Card manager > Card > Profile**), click **Update** to display the Card Profile Description window.
- 2 Select the service implementations that are associated with this card profile and move them to the **Selected services** list:

Figure 17 - Service Implementations Associated with a Card Profile



- 3 Use the left and right arrow buttons to move all the service implementations that it is possible to request for this card type into the **Selected services** list.

To be listed in the available services list, a service implementation must have been provisioned in the Card Manager database and specified as being Active. See “Chapter 5 - Managing Services” for details.

When you have finished, click **Update**. When issuing service requests or running campaigns targeted at cards based on this card profile, only services in the **Selected services** list are available for selection. See “Chapter 8 - Submitting Service Requests” and “Chapter 9 - Managing Campaigns” for details.

Defining Security Properties

Security properties are defined separately in the card profile for each entity (Java applet, or applet instance) on the card. Each entity's properties include:

- General properties, such as the application's identifier (AID), version number and identifying label.
 - One or more *security levels* describing the security mechanisms used by the application.

Correctly defining the entities' security properties is fundamental to good security management in OTA Manager V5.1. The properties can be defined either by batchloading them as part of the card profile or, for native card folders and applications only, using the management interface.

For an overview of the security mechanisms in OTA Manager V5.1, refer to “Appendix H - The “Security Domain” Security Model”.

Batchloading Entity Definitions

The card profile must include definitions of all the application entities on the card and their related security settings.

The applications can include:

- Native applications and Directory Files (DFs), dedicated to native cards.
 - Java packages, applets, and instances, for Java cards.

Typically, the list includes interpreter applets responsible for managing the contents of the cards, for example:

- RAM applets, for example, the GOP Interpreter system applet on GSM cards, that process and execute OTA commands destined for applets.
 - RFM applets, for example, the GSM Interpreter system applet on GSM cards that manage the files and data in the card's GSM file system.

You batchload these definitions using the **EntityProperties** element and its sub-elements:

- **JavaPackageProperties**, **JavaAppletProperties**, and **JavaAppletInstanceProperties** for Java cards
 - **DirectoryFileProperties**, **NativeApplicationProperties**, and **NativeServiceProperties** for native cards.

Java Card Properties

The following XML shows an example of a Java applet instance:

```
EntityProperties>
<JavaAppletInstanceProperties
    instanceAid="A000000018434D08090A0B0C000000"
    appletAid="A000000018434D08090A0B0C000000"
    packageAid="A000000018434D"
    packageMinorVersion="0"
    packageMajorVersion="1"
    label="RAM applet instance" >
<SecurisationSet>
    <Securisation0348 index="0" spi="0000" kic="00" kid="00" />
</SecurisationSet>
</JavaAppletInstanceProperties>
...

```

```
</EntityProperties>
```

Where:

- The **instanceAid**, **appletAid**, and **packageAid** attributes define the AIDs of the applet instance, applet, and package respectively.
- The **packageMinorVersion** and **packageMajorVersion** elements contain the minor and major version numbers of the package.
- The values of the **label** attribute is used to generate the list of applications displayed on the **Properties** property sheet as the **Application list** window (see "Figure 20" on page 53).
- The **SecurisationSet** element is used to define the security mechanisms implemented in the message (and its contents therefore depends on the target card type). It is described in more detail in "Batchloading Security Settings" on page 51.

The following example shows an applet:

```
<JavaAppletProperties appletAid="A00000001803090000000000B00010"  
    packageAid="A000000018210008" packageMinorVersion="0"  
    packageMajorVersion="1" label="RFM applet"  
    concatenationAvailable="true" isSelectable="true">  
</JavaAppletProperties>
```

Where:

- The **concatenationAvailable** option specifies whether the applet supports GSM 03.40 concatenation.
- The **isSelectable** option indicates whether the applet entity can be selected for use.

CAT-TP Compliant Java Card Properties

The following example shows a CAT-TP compliant Java applet instance:

```
<EntityProperties>  
    <JavaAppletInstanceProperties  
        instanceAid="A000000018434D08090A0B0C000000"  
        appletAid="A000000018434D08090A0B0C000000"  
        packageAid="A000000018434D"  
        packageMinorVersion="0"  
        packageMajorVersion="1"  
        label="RAM applet instance"  
        cattpPort="10">  
        <SecurisationSet>  
            <Securisation0348 index="0" spi="0000" kic="00" kid="00" />  
        </SecurisationSet>  
    </JavaAppletInstanceProperties>  
    ...  
</EntityProperties>
```

Where:

- The **instanceAid**, **appletAid**, and **packageAid** attributes define the AIDs of the applet instance, applet, and package respectively.
- The **packageMinorVersion** and **packageMajorVersion** elements contain the minor and major version numbers of the package.

- The value of the **label** attribute is used to generate the list of applications displayed on the Properties property sheet as the Application list window (see “Figure 20” on page 53).
- The value of the **cattpPort** attribute is the CAT-TP port number used to establish the CAT-TP connection with the SIM card. This value is optional.
- The **SecurisationSet** element is used to define the security mechanisms implemented in the message (and its contents therefore depends on the target card type). It is described in more detail in “Batchloading Security Settings” on page 51.

To batchload entity definitions:

- 1 Refer to “Chapter A - Batchloading XML Files” for a sample card profile file.
- 2 Batchload the XML file on the Card Profile list window (**Card manager > Card > Profile**).

Batchloading Security Settings

Security settings can be batchloaded for the following entity types:

- Native applications
- Java Packages
- Java Applets
- Java Applet Instances

To batchload security settings:

- 1 Refer to “Chapter A - Batchloading XML Files” for a sample card profile batchload file.
- 2 Batchload the XML file on the Card Profile list window (**Card manager > Card > Profile**).

In the XML, security settings are defined within the **SecurisationSet** element. Within this element, add the appropriate security elements and attributes:

- **SecurisationNative** for native applications
- **SecurisationEsmsV1** for ESMSV1
- **SecurisationEsmsV2** for ESMSV2
- **Securisation0348** for Java applications.

The attributes of each element differ. Refer to the examples in “Appendix H - The “Security Domain” Security Model” for details.

GSM 03.48 Security Settings

For GSM 03.48 applications, the **Securisation0348** element requires the following attributes:

```
<SecurisationSet>
    <Securisation0348 index="4" spi="1E21" kic="10" kid="10"/>
    <Securisation0348 index="3" spi="1621" kic="10" kid="10"/>
    <Securisation0348 index="2" spi="0621" kic="10" kid="10"/>
    <Securisation0348 index="1" spi="0221" kic="00" kid="10"/>
    <Securisation0348 index="0" spi="0000" kic="00" kid="00"/>
</SecurisationSet>
```

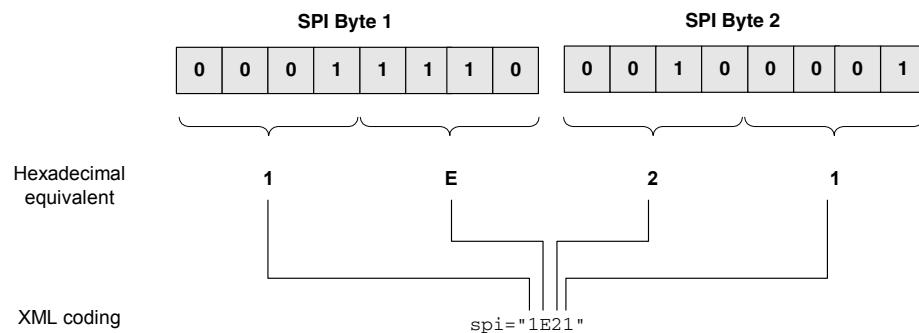
Where:

- **index.** The security level to assign to the security settings. The security level must be unique among all sibling **Securisation0348** elements of an application.
- **spi.** The value of the two SPI bytes to insert in the GSM 03.48 command header for all services using this security level.
- **kic.** The value of the KiC byte to insert in the GSM 03.48 command header for all services using this security level.
- **kid.** The value of the KID byte to insert in the GSM 03.48 command header for all services using this security level.

To encode these elements:

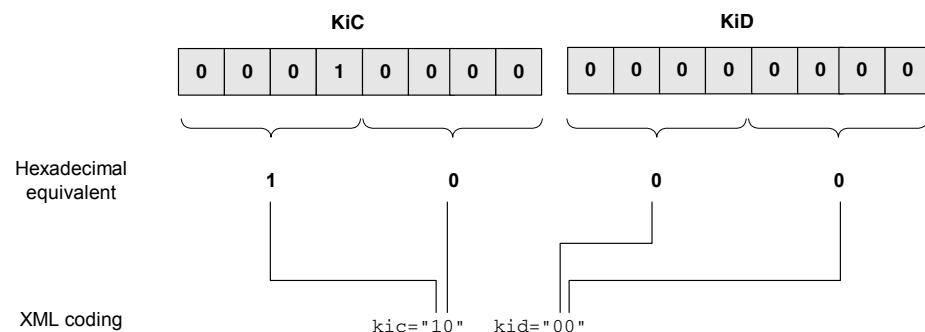
- 1 Refer to “Appendix H - The “Security Domain” Security Model” for examples showing the binary coding of the SPI, KiC, and KID bytes.
- 2 Translate the SPI binary values to their hexadecimal equivalent, as shown below:

Figure 18 - Encoding the SPI Bytes



- 3 Translate the KiC and KID binary values to their hexadecimal equivalent, as shown below:

Figure 19 - Encoding the KiC and KID Bytes



Tip: You can use the same technique in reverse to decode the hexadecimal encodings of SPI, KiC and KID values that appear in log files.

ESMSV1, ESMSV2 and Native Security Settings

An ESMS V1 example:

```
<SecurisationSet>
    <SecurisationEsmsV1 index="1" system fileId="7F10"
        certifKeyNumber="01" cipherKeyNumber="00"/>
</SecurisationSet>
```

The **systemFileId** must correspond to the Login DF, the **certifKeyNumber** must correspond to an existing key in the EF Key file. The **cipherKeyNumber** is generally set to "00" because there is only a small number of these cards with ciphering capabilities in the field.

An ESMS V2 example:

```
<SecurisationSet>
  <SecurisationEsmsV2 index="2" app="56" sec="07"
    certifKeyNumber="01" cipherKeyNumber="00"/>
</SecurisationSet>
```

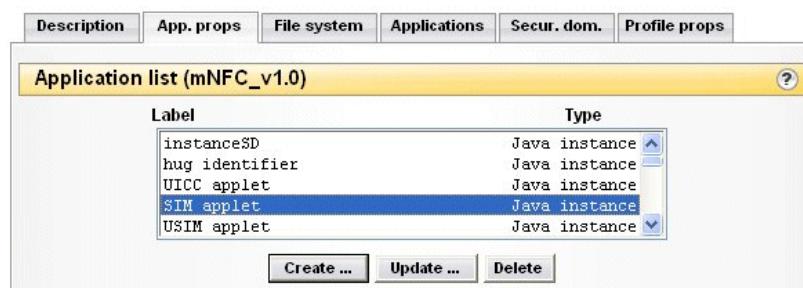
Refer to "Appendix H - The "Security Domain" Security Model" for details on the XML elements and attributes.

Updating Security Settings with the Management Interface

Displaying the Application List

To display a list of the applications that have been provisioned on the card, display the Card profile description window and choose the **App. props** property sheet:

Figure 20 - The Application List



The **Application list** includes:

- Java packages, applets, and instances.
- Directory files
- Native applications

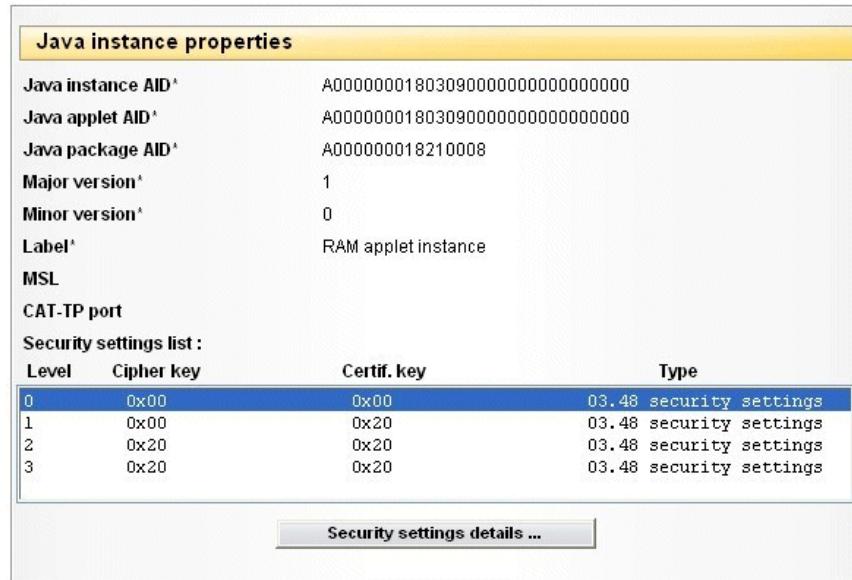
Any Java applications in this list must have been defined in the XML file that was used to batchload this card profile.

From this window, you can view, update and delete the applications' properties.

Java packages, applets, and instances can only be added to this list:

- By batchloading a card profile
- Automatically, whenever an applet is successfully downloaded to at least one card associated with this card profile.

To view the properties of a Java applet instance, for example, select a **Java instance** item in the Application list window. A properties window appears below the list of applications, for example:

Figure 21 - Applet Instance Properties

The properties window shows:

- The AIDs of the applet instance, the applet from which it was instantiated, and the package to which the applet belongs.
- The **Major version** and **Minor version** numbers of the package from which the instance is created.
- The **Label** of the instance. This value is specified in the XML file when batchloading the card profile (see “Batchloading Entity Definitions” on page 49).
- For Java cards that support the parameter, the minimum security level (**MSL**) required by the applet. See “Appendix H - The “Security Domain” Security Model” for details.
- For CAT-TP compliant Java cards, the **CAT-TP port** number. The value must be between 1 and 1023.
- Security level settings for the applet instance.

From this window, you can also add security settings to, or modify and delete security settings from an applet’s properties. This can also be done by batchloading the security settings.

Adding Security Settings

Each application can be associated with a different set of security parameters in the Card Settings database. These parameters define the level of security that OTA Manager V5.1 is to use when communicating with the application.

Native applications and DFs use ESMS V1 or ESMS V2 security settings.

Java packages, applets, and instances use GSM 03.48 security settings.

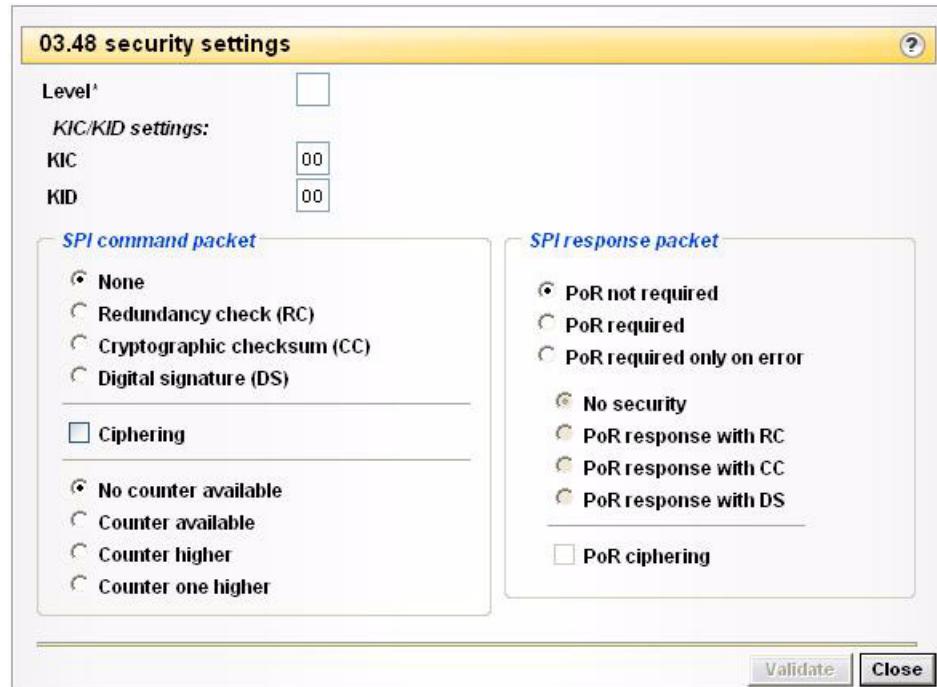
To add security settings for an application, click the **Add new security settings** button on the application’s property window.

For example, to create GSM 03.48 security settings:

- 1 On the **Card profile list** window, select the profile and click **Update**.
- 2 On the **Card profile description** window, click the **App. Props** tab.
- 3 On the **Applications** window, select the properties file and click **Update**.

- 4 On the **File properties** window, click **Add security item**:

Figure 22 - GSM 03.48 Security Settings



- 5 On the 03.48 Security Settings window, enter values for parameters shown in the following table and click **Validate**.

Table 4 - GSM 03.48 Security Mechanisms

Parameter Name	Description
Security Level	The security level used as the identifier of security.
KiC/KID settings area	
KiC	The key set number to use for ciphering. Valid values are 0 to 15.
KID	The key set number to use for certificate computation. Valid values are 0 to 15.
SPI command packet area	
None Redundancy check (RC) Cryptographic check (CC) Digital signature (DS)	Select the security mechanisms required by the application to authenticate the message: redundancy check, cryptographic checksum, or none. Digital signatures are not currently supported. See "Appendix H - The "Security Domain" Security Model" for descriptions of the security mechanisms.
Ciphering	Whether message ciphering is required to guarantee the confidentiality of the message's contents.
No counter available Counter available Counter higher Counter one higher	Select requirements for the use of a synchronization counter to prevent replay attacks on the card: counter available, counter higher, counter one higher, or no counter available. See "Appendix H - The "Security Domain" Security Model" for a description of the synchronization counter.

Table 4 - GSM 03.48 Security Mechanisms (continued)

Parameter Name	Description
SPI response packet area	
PoR not required	Select the Proof of Receipt settings, which enable you to know if the service was correctly executed on the card.
PoR required	See "Handling the Proof of Receipt" on page 286.
PoR required only on error	
No security	Specify either no security, PoR response with redundancy check,
PoR response with RC	PoR response with cryptographic check, or PoR response with digital signature. Digital signatures are not currently supported.
PoR with CC	
PoR response with DS	
PoR ciphering	Specify whether PoR ciphering is required.

Defining the Card's Structure

A card's structure is made up of

- Internal file system: directory files, elementary files (linear, cyclic, and transparent). Properties relate to the different attributes of the file's structure; for example, record number, record size and so on.
- Applications: native applets and services, Java package files, Java applets, and Java applet instances

Specifying the Contents of Mapped Files

In 3G cards, a number of SIM and USIM files can be *mapped*, that is, separate 2G and 3G versions of the files exist, but one is mapped onto the other so that both share the same contents.

To specify that a file's contents are shared, the size of the mapped 3G file must be set to zero in the card profile. Then, when the 3G version of a service is launched that updates the contents of the file, the contents of the 2G version of the file are updated and the changes are automatically reflected in the mapped 3G version.

The mechanism is therefore as follows:

- If a 3G file size is equal to zero, update the 2G file's contents (and reflect the changes in the 3G file).
- If a 3G file size is not equal to zero, update the 3G file's contents only.

For the standard services delivered with OTA Manager V5.1, the `LinkFile.properties` file contains a list of mapped files and `CustomLinkFile.properties` contains a list of mapped files for custom services. Both files are located in the `$RCA_PLUGIN/com/gemplus/gcota/card` folder.

This mapping mechanism only applies to card contents in which the files are normally updated.

Batchloading Card Structure Elements

The card structure is placed between **CardStructure** elements in the card profile.

The value of the **name** attribute of the element must match that of the **name** attribute of the **CardProfile** element:

```
<CardProfile ... name= "GX3G_v2.2_128K">
...
<CardStructure name="GX3G_v2.2_128K">
```

```
    ...
  </CardStructure>
</CardProfile>
```

Some examples of individual structure elements follow.

A Master File 3F00:

```
<CardFile identifier="3F00" name="MF" type="0" fileSize="6474">  
</CardFile>
```

The Dedicated File 7F10:

```
<CardFile identifier="7F10" type="&DF_FILE;" name="TELECOM" type="1"  
fileSize="6336">  
</CardFile>
```

An Elementary File 6F40 (EF_{MSISDN}), with default binary contents specified:

Note that when the default content of a file is specified (**defaultData** attribute), the number of bytes must be equal to the product of the **recordNumber** and **recordSize** attribute values ($2^8=56$ in this case).

For 3G card profiles, it is possible to represent the USIM ADF (Application Dedicated File) by creating a DF with the identifier “7FFF”. If any other value is specified, file interpretation and card content updates are not possible.

A Java package:

```
<JavaPackage aid="A0000000180309"  
    majorVersion="1" minorVersion="0"  
    estimatedSize="0"  
    securityDomain="A000000018434D08090A0B0C000000">  
    ...  
</JavaPackage>
```

A Java applet with one instantiated instance:

```
<JavaApplet aid="A00000001803090000000000B00010">
  <JavaAppletInstance aid="A00000001803090000000000B00010"
    status="SELECTABLE" estimatedSize="0"
    securityDomain="A000000018434D08090A0B0C000000" tar="B00010"/>
</JavaApplet>
```

Where:

- The **tar** attribute value is the TAR of the applet. For cards that support it, this attribute can also contain a list of TARs, separated by a semi-colon (";").
The TAR is used to target the applet for OTA messages to be executed by the applet. For this reason, the TAR specified in the service implementation must equal one of the TARs defined in the card structure area of the card profile. Refer to "Chapter 5 - Managing Services".
 - The **securityDomain** attribute value is a reference to the location of the keys used for ciphering and cryptographic checksum calculations.

Its value must match that of a security domain declared in the card security file:

- For Java cards, the attribute value must match the AID of the on-card security domain entity responsible for managing keys and security computations, for example `securityDomainID="A000000018434D"`.
- For native cards, the attribute value identifies the file containing the keys, for example, `securityDomain="3F00"`.

Refer to “Transport Keys” on page 77 for details.

Each `JavaAppletInstance` declaration in the `CardStructure` section of the card profile file must also include a reference to a `JavaAppletInstanceProperties` element in the `EntityProperties` area of the card profile:

```
<EntityProperties>
  <JavaAppletInstanceProperties
    instanceAid="A000000018030900000000000B00010" ...
  ...
</EntityProperties>
...
<CardStructure>
...
  <JavaAppletInstance aid="A000000018030900000000000B00010"
    status="SELECTABLE" estimatedSize="0"
    securityDomain="A000000018434D08090A0B0C000000" tar="B00010"/>
...
</CardStructure>
```

This mechanism is described in detail in “Appendix H - The “Security Domain” Security Model”.

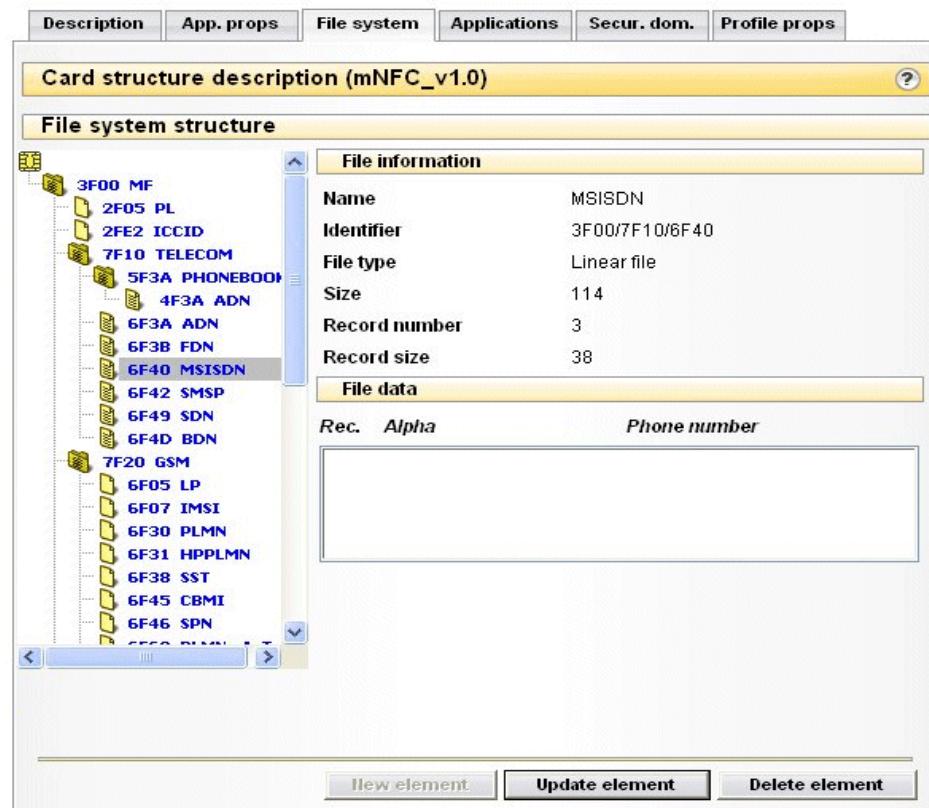
Refer to “Appendix A - Batchloading XML Files” for details on the coding of individual elements.

Defining the File System with the Management Interface

To define the elements that make up a card’s structure:

- 1 On the **Card profile description** window, choose the **File system** property sheet. A graphical representation of the card’s file system is displayed:

Figure 23 - The File System Window



The following element types may be displayed:

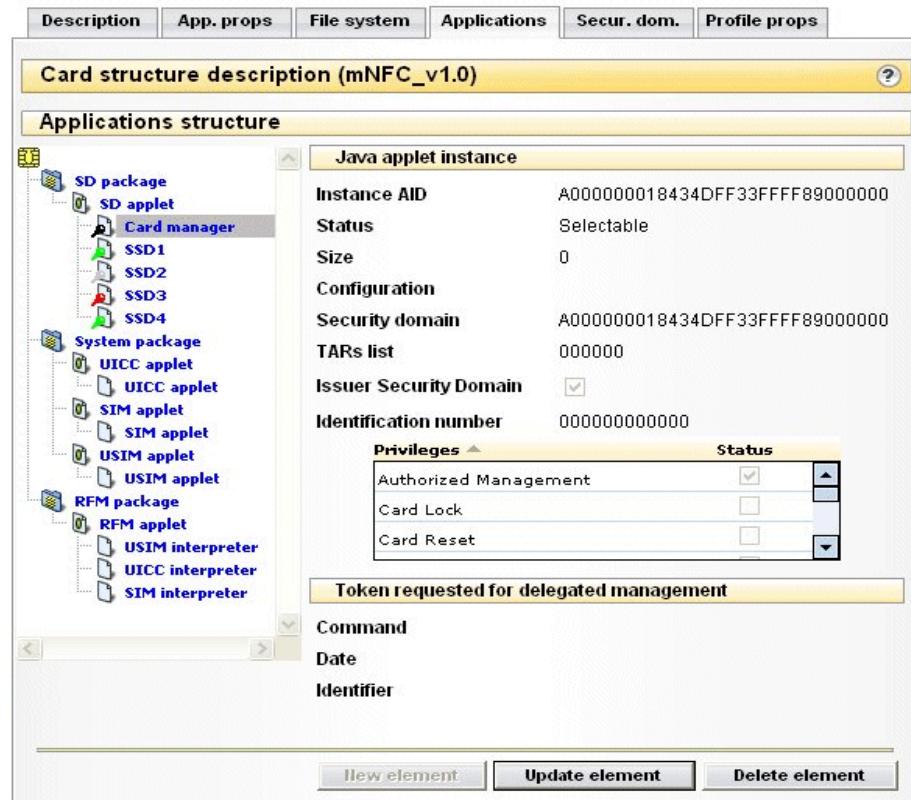
- The root of the card structure
- The Master File that is the root of the file system. This is a unique, root directory of the file system that contains all the elementary files required for the card structure.
- Directory File (DF).
 - Linear (LF): Binary/ASCII files with records of equal size.
 - Cyclic (CF): Binary/ASCII files with a series of records of equal size, and in cyclic order that do not specify a start or end line. When the file is full, the latest record in the file is overwritten in order for more records to be created.
 - Transparent (TF): Binary or ASCII files without a record structure.
- A native application folder

Click for more information on the properties of each element type and how to add new elements.

Defining Application Structure with the Management Interface

To define the applications that make up a card's structure:

- 1 On the **Card profile description** window, choose the **Applications** property sheet. A graphical representation of the card's applications is displayed:

Figure 24 - The Applications Structure Window

The following element types are displayed in the structured view:

- The root of the card structure.
- A Java package present in the card profile content.
- An application, for example “SIM applet”.
- An application instance, for example “UICC Applet”.
- The Issuer Security Domain (ISD), for example “Card Manager”. An ISD is always the first security domain installed on the card.
- A security domain instance.
- A security domain instance with the Authorized Management privilege.
- A security domain instance with the Delegated Management privilege.

Click for more information on the properties of each element type and how to add new elements.

Deleting Elements from the Card Structure

Because all card instances are based on a card profile, deleting or updating the card’s structure automatically invalidates all card instances based on it. You should therefore only delete elements from new card profiles (for example, when creating a new card profile by copying an existing one).

To delete an item from the card structure:

- 1 Edit the card profile by clicking **Update** on the Card Profile list window (see “Figure 16” on page 45).
- 2 Select the **File system** or **Application** property sheet.
- 3 Select the item to delete in the structure view.
- 4 Click **Delete element**. A warning message appears warning you against modifying this card profile if service requests or campaigns based on it are currently running. Click **OK** to continue.

Defining Security Domain Structure with the Management Interface

To define the structure of security domains:

- 1 On the **Card profile description** window, choose the **Security domain** property sheet. A graphical representation of the card’s security domains is displayed:

Figure 25 - The Security Domain Structure Window



The following elements of a security domain’s structure are displayed:

- The root of the card structure.
- A Java package present in the card profile content.
- An application, for example “SIM applet”.

-  An application instance, for example “UICC Applet”.
-  The Issuer Security Domain (ISD), for example “Card Manager”. An ISD is always the first security domain installed on the card.
-  A security domain instance.
-  A security domain instance with the Authorized Management privilege.
-  A security domain instance with the Delegated Management privilege.

Defining Card Profile Properties

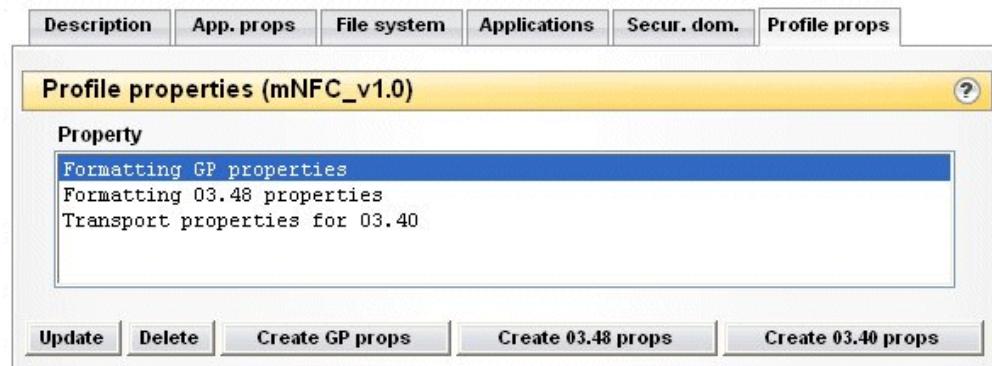
A card profile contains details on how SMS messages sent to the card are to be formatted. The following property types can be specified:

- 1 GlobalPlatform (GP) formatting properties
- 2 03.48 formatting properties
- 3 03.40 transport properties.

To view card profile properties:

- 1 On the **Card profile description** window, choose the **Profile props** property sheet. The following window is displayed:

Figure 26 - The Profile Properties Window



Defining GlobalPlatform Formatting Properties

Card profile parameters ensure correct formatting with older card types and interoperable cards from other manufacturers.

To batchload GlobalPlatform formatting properties:

In the XML, GlobalPlatform formatting properties are defined within the **FormattingProperties** element. For example:

```
<FormattingGPProperties>
  ...
  <FormattingGPProperties
    multipleLoadCommandPerMessage="true"
    maxLoadCommandLength="255" >
    tokenKeyVersion="70"
    tokenKeyIndex="1"
    receiptKeyVersion="71"
    receiptKeyIndex="1">
```

```
</FormattingGPProperties>
```

Where:

- **multipleLoadCommandPerMessage**. Whether or not a command packet contains only a single **Load** command. This check box is selected by default (“true”).
- **maxLoadCommandLength**. The maximum size of a **Load** command, in bytes. The default and maximum value is 255 (bytes).
- **tokenKeyVersion**. The key version of the key to use for token generation. The default value is 70. The value must be between 1 and 127.
- **tokenKeyIndex**. The key index of the key to use for token generation. The default value is 1. The value must be between 1 and 127.
- **receiptKeyVersion**. The key version of the key to use for receipt verification. The default value is 71. The value must be between 1 and 127.
- **receiptKeyIndex**. The key index of the key to use for receipt verification. The default value is 1. The value must be between 1 and 127.

To define GlobalPlatform formatting properties in the management interface:

- 1 On the **Card profile description** window, choose the **Profile props** property sheet.
- 2 Click **Create GP props**. The **Global Platform Formatting Properties** window is displayed:

Figure 27 - Global Platform Formatting Properties Window



- 3 Specify values of GlobalPlatform formatting properties as required.
- 4 Click **Update card profile**.

Defining 03.48 Formatting Properties

OTA Manager V5.1 is fully compliant with the ETSI GSM 03.48 standard. However, the GSM 03.48 standard is open to interpretation in some respects, so OTA Manager V5.1 allows you to configure certain aspects of message formatting, enabling it to be used to format and unformat messages for both Gemalto and other manufacturers' interoperable cards.

To batchload GSM 03.48 formatting properties:

In the XML, GSM 03.48 formatting properties are defined within the **FormattingProperties** element. For example:

```
<FormattingProperties>
  <Formatting0348Properties
    isCplChlRequiredInCertificate="true"
    rpFieldRequiredInCertificate="RP_CERTIF_ALL">
  </Formatting0348Properties>
```

...
 </FormattingProperties>

Where:

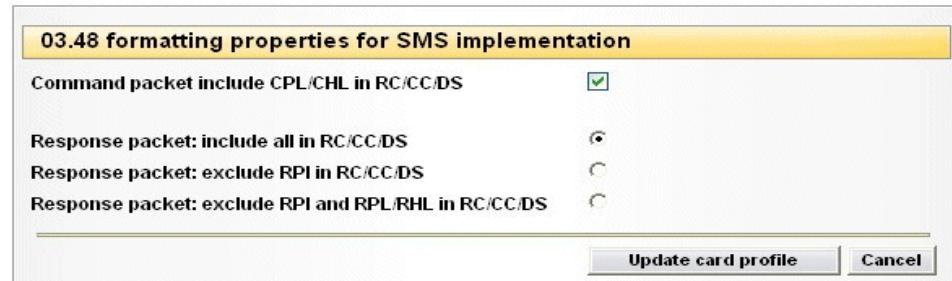
- **isCplChlRequiredInCertificate.** Allows you to specify how the platform handles calculation of the RC or CC for a command packet. Possible values are:
 - **True** (or attribute omitted): Include CPL, CHL, SPI, KiC, KID, TAR, CNTR, PCNTR, and DATA fields when calculating the certificate (RC or CC) for a command packet.
 - **False.** Include only SPI, KiC, KID, TAR, CNTR, PCNTR, and DATA fields when calculating the certificate (RC or CC) for a command packet.
- **rpFieldRequiredInCertificate.** Allows you to specify how the platform handles calculation of the RC or CC for a response packet. Possible values are
 - **RP_CERTIF_ALL** (or attribute omitted). Include RPI, RPL, RHL, TAR, CNTR.PCNTR, SC, and DATA fields when calculating the certificate (RC or CC) for a response packet.
 - **RP_CERTIF_WITHOUT_RPI.** Include RPL, RHL, TAR, CNTR.PCNTR, SC, and DATA fields when calculating the certificate (RC or CC) for a response packet.
 - **RP_CERTIF_WITHOUT_RPI_RPL_RHL.** Include TAR, CNTR.PCNTR, SC, and DATA fields when calculating the certificate (RC or CC) for a command packet.

Refer to “Appendix I - GSM 03.48 Message Structure” for details.

To define GSM 03.48 formatting properties for SMS messages in the management interface:

- 1 On the **Card profile description** window, choose the **Profile Props** property sheet.
- 2 Click **Create 03.48 props.** The **0348 formatting properties** window is displayed:

Figure 28 - 03.48 Formatting Properties Window



- 3 Specify values of 03.48 formatting properties as required.
- 4 Click **Update card profile**.

Defining 03.40 Transport Properties

For SIM cards that support the short message concatenation feature described in the GSM 03.40 (3GPP TS 23.040) standard, OTA Manager splits formatted OTA messages into GSM 03.40 compliant packets. To ensure compatibility with all SIM cards, OTA Manager allows you to specify the buffer size and concatenation buffer space of SIM cards.

In the XML, transport properties are defined within the **TransportProperties0340** element. For example:

```
<TransportProperties>
  <TransportProperties0340
    tpudBufferSize="140"
    concatBufferSpace="255" >
  </TransportProperties0340>
  ...
</TransportProperties>
```

Where:

- **tpudBufferSize**. The maximum size of TP-User Data field that can be handled by the card, which also corresponds to the maximum size of a single SMS message that can be handled by the card. The default and maximum value is 140 (bytes)
- **concatBufferSpace**. The maximum size of the card's concatenation buffer, expressed in terms of a number of SMS messages. The minimum value is 1 (if the card does not handle concatenation at all) and the maximum value 255 (the limit imposed by GSM 03.40). The default value is 255.

If the “Concatenation” option is ticked on the RAM Java Applet Properties window, the **concatBufferSpace** parameter must be set equal to the maximum number of SMS messages that can be sent for this card profile.

If the “Concatenation” option is unchecked, the **concatBufferSpace** parameter must be set to 1.

To define GSM 03.40 transport properties in the management interface:

- 1 On the **Card profile description** window, choose the **Profile Props** property sheet.
- 2 Click **Create 03.40 props**. The **03.40 transport properties** window is displayed:

Figure 29 - 03.40 Transport Properties Window



- 3 Specify values of 03.48 formatting properties as required.
- 4 Click **Update card profile**.

Managing Card Instances

This chapter describes how to:

- Provision and manage groups of cards.
- Provision and manage card instances.
- Provision and manage the Card Security database.
- Define card operating systems, Java virtual machines, card definitions, and card vendors.

Introduction

When a batch of SIM cards is manufactured, all the cards have an identical electrical profile. This electrical profile is represented in OTA Manager V5.1 by means of a *card profile*.

When the cards are personalized prior to being distributed to subscribers, each card must be provisioned within OTA Manager V5.1's Card Manager database as a *card instance*. A card instance is simply an image of the contents of a single card dedicated to a single subscriber.

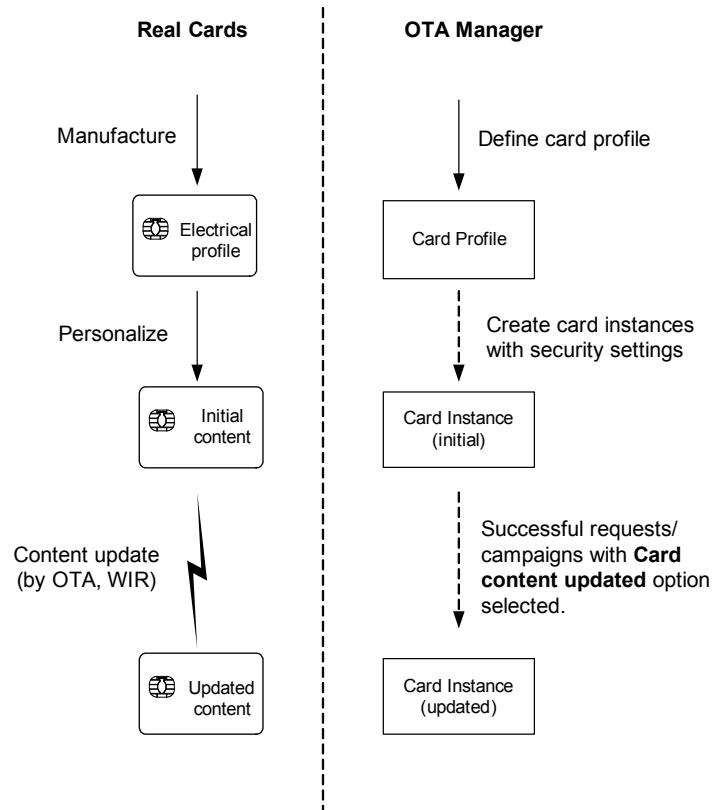
Each card instance contains:

- A reference to the card profile upon which it is based.
- Information required to uniquely identify the card and the subscriber to whom it is attributed: the ICCID (serial number) of the card, IMSI (mobile identification number for the network), MSISDN (phone number), and so on.
- Security settings, such as key values and synchronization counter values.

The card instance stored in the Card Manager database is only updated by OTA Manager V5.1 when the following conditions are satisfied:

- 1 A service request or campaign is successful.
- 2 The **Card content updated** option is selected in the service implementation options. See "Chapter 5 - Managing Services" for details.

This situation is illustrated in "Figure 30" on page 68.

Figure 30 - Card Content in Real Cards and OTA Manager V5.1

Because the security settings are sensitive, they are stored in a separate dedicated database, the Card Security database. The contents of this database are protected in several ways:

- The database cannot be accessed through the management interface.
- The contents of the Card Security database are themselves ciphered.

All operations on the Card Security database can be performed independently of the Card Manager database. However, both databases must be correctly provisioned in order for OTA Manager V5.1 to function correctly.

Creating and Managing Groups

Note: You can manage groups only when the **Group Management** option has been specified in your user profile.

A group is a set of SIM cards that have certain criteria in common. For example, you could define a group of “Gold Card” users, or a group of users that have a specific handset type, for example, “3G users”.

Group management allows you to divide cards into more easily manageable segments. You use group definitions when defining targets for campaigns, or when searching for card instances.

Each group comprises the following:

- The group ID; a unique identifier of the group used as the key field in searches.
- A free-text description of the group content
- Creation date and last modification date of the group

- Current status
- A list of the card instances that are assigned to the group.

Creating a group definition can be done either through batchloading or with the management interface. OTA Manager then creates the group in asynchronous mode. This process can be time-consuming, so large groups are best created overnight. It is recommended to limit the number of groups to around 10 (the upper limit is fixed by the **GROUPS_NBR** Card Manager installation parameter).

While group creation is being processed, the group's status is set to OPERATION IN PROGRESS. The result is stored in a report file that contains, for each new card in the group, the execution status (FAILED or SUCCEEDED).

If adding cards is successful, the group's status changes to OPERATIONAL.

If a create or update operation on a group fails, or if a group is found in the OPERATION_IN_PROGRESS state when the Card Manager starts up, the platform assigns a CORRUPTED status to the group. When a group is CORRUPTED, you can:

- If you decide that the group's definition is correct, manually change the status to OPERATIONAL
- Repeat the modifications to the group
- Delete the group and recreate it.

If the execution status is FAILED, a detailed error message is also available. The import process generates an error in the following cases:

- The card instance is not present in the Card Manager database.
- A card instance is specified more than one time in an input file. In this case, only the first occurrence is taken into account. All other occurrences are rejected.

Batchloading Group Definitions

To batchload group definitions:

- 1 Create an XML file containing details of the group to add. See “Appendix A - Batchloading XML Files” for details on the format of the group definition file.

The following is an example:

```
<CardGroupList>
<CardGroup id="GroupGXXV3"
           description="Group with 20 cards GXXV3">

    <!-- Targets defined by their ICCID values.          -->
    <Card ICCID="000000001600900601"/>
    <Card ICCID="000000001600900604"/>
    <Card ICCID="000000001600900607"/>
    <Card ICCID="000000001600900610"/>
    <Card ICCID="000000001600900613"/>
    <Card ICCID="000000001600900616"/>
    <Card ICCID="000000001600900619"/>
    <!-- Targets defined by their IMSI values.          -->
    <Card IMSI="260091234500603"/>
    <Card IMSI="260091234500606"/>
    <Card IMSI="260091234500609"/>
    <Card IMSI="260091234500612"/>
    <Card IMSI="260091234500615"/>
    <Card IMSI="260091234500618"/>
```

```

<!-- Targets defined by their MSISDN values. -->
<Card MSISDN="33600900602"/>
<Card MSISDN="33600900605"/>
<Card MSISDN="33600900608"/>
<Card MSISDN="33600900611"/>
<Card MSISDN="33600900614"/>
<Card MSISDN="33600900617"/>
<Card MSISDN="33600900620"/>
</CardGroup>
</CardGroupList>

```

Notice that you can use a mixture of **MSISDN**, **ICCID** and **IMSI** attributes to identify the cards to add to the group.

- 2 On the Group Management window (**Card manager > Card > Group management**), click **Batchload**. Click  for details.

Creating Group Definitions in the Management Interface

To work with card groups in the management interface:

- 1 Do either of the following:
 - On the **Welcome** window, click **Card manager**
 - Click **Card** on any management interface window.
- 2 On the left frame of the Card management window, click **Card > Group management**. This displays all available groups in the platform. From this window, you can perform the following tasks:
 - Create a new group. There are two steps to creating a group:
Creating a group description.
Selecting card instances to assign to the group. This can be done either by searching for card instances or by importing a list of card instances.
 - View a specific group's details.
 - Update a specific group.
 - Delete an existing group from the platform.
 - Export an existing group to a specified file.

Click  for details.

Managing Card Instances

The following provides information on creating and managing card instances in the Card Manager database.

Card instances can be created either by batchloading or through the management interface. While it is easy to create and manage individual card instances through the management interface, batchloading is recommended whenever you need to create multiple card instances.

The general procedure is:

- 1 Create, or search for and select, card instances. You can search for a card instance using card search criteria such as the MSISDN of a card instance or by an applet that is, or is not, loaded to a specific card instance using the applet search criteria.

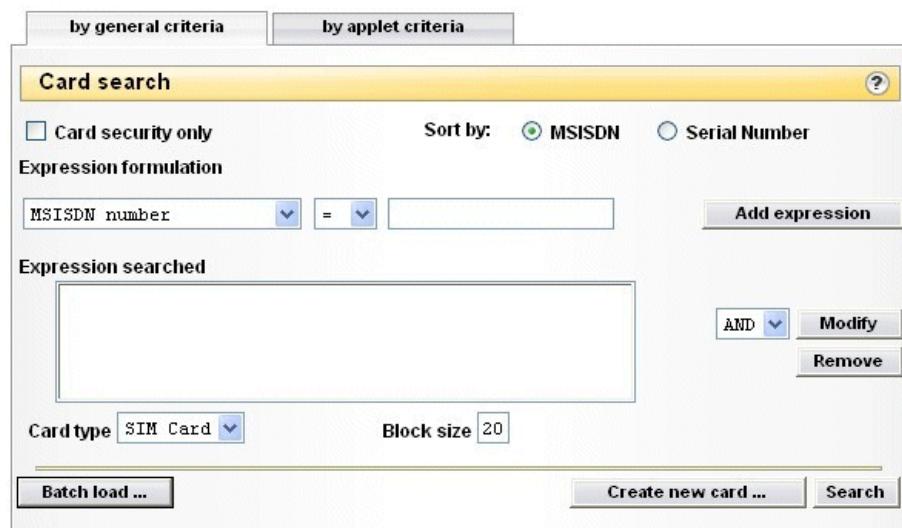
At this stage, you can instead choose to batchload the information. See “Batchloading Card Instances” on page 72 for details.

- 2 Manage the contents of a selected card instance. You can:
 - Update certain card properties (for example, its current state or whether it is a linked card)
 - Delete card instances
 - View details of the current card contents
 - Update the security settings associated with the card contents.

To work with card instances in the management interface:

- 1 Do either of the following:
 - On the **Welcome** window, click **Card manager**
 - Click **Card** on any management interface window.
- 2 On the left frame of the Card management window, click **Card > Instance**. The Card Search window is displayed:

Figure 31 - The Card Search Window



On this window, you can:

- Click **Batch load** to load card content definitions or security settings from XML files. See “Batchloading Card Instances” that follows.
- **Search** for existing card instances using card search or applet search criteria.
- Click **Create new card** to create new card instances using the management interface.

Click  for details.

Batchloading Card Instances

Card instance definitions can be batchloaded from XML files directly into the Card Manager database.

To batchload card instances:

- 1 Prepare an XML file defining the card instances. Refer to “Appendix A - Batchloading XML Files” for the definition of the card content DTD. An example XML file follows:

```
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE CardContentModule SYSTEM "cardframework.dtd">
<CardContentModule>
  <SimCard
    serialNumber="892300456333378610"
    state="1"
    imsi="208200509003050"
    msisdn="0613113709"
    profileName="GX3G"
    checkSecurity="true"/>
  <SimCard
    serialNumber="892300456333378611"
    state="1"
    imsi="208200509003051"
    msisdn="0613113710"
    profileName="GX3G"
    checkSecurity="true"/>
</CardContentModule>
```

Refer to “Card Instances” on page 157 for details on the individual fields.

Note in particular that:

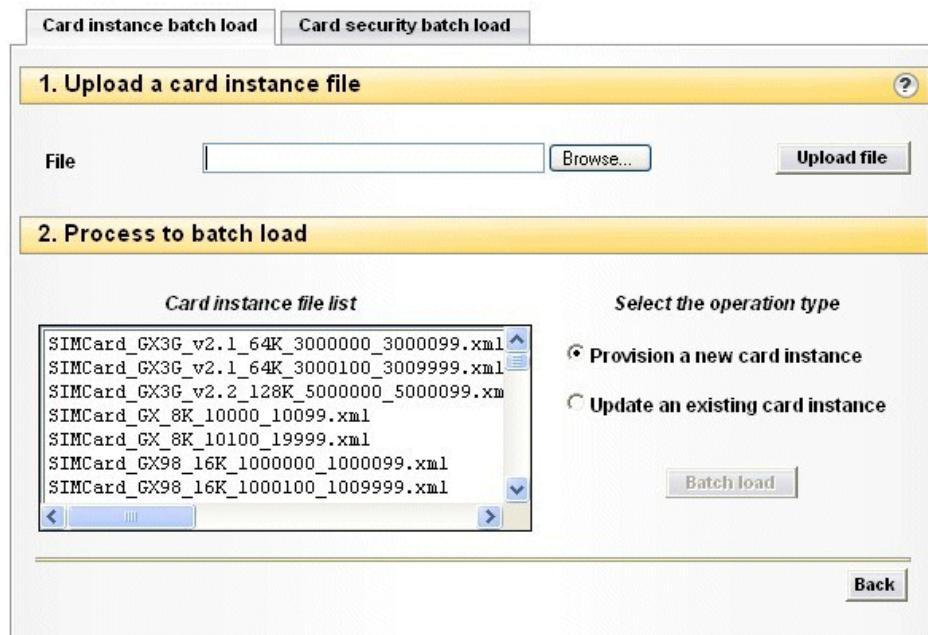
- The card profile reference on the **profileName** attribute is mandatory and the card profile to which it refers must have been provisioned previously.
- The **msisdn** attribute is the phone number of the card, coded on up to 10 digits. The number can be prefixed with a “+”, in which case service requests will be prefixed with the appropriate international TON/NPI.
- The **iccid** attribute (not shown in this example) is the serial number of the card, coded on up to 20 digits.
- The **imsi** attribute is the unique network identifier of the card, coded on 15 digits.
- The **checkSecurity** attribute, when set to “true”, checks that the card security database contains the necessary security for the selected card is set. If no data is referenced in the database, the card provisioning fails and an error is generated.

If the value of this attribute is missing or set to “false”, no security checks are done or the value is not defined,

- The card instance file can also include details of any applications that have been added to the card instance in addition to those defined for the card profile. Refer to “Appendix A - Batchloading XML Files” for more information.

- 2 On the **Search Card Content** window (see “Figure 31” on page 71), click **Batch load**. The following window is displayed:

Figure 32 - Batchloading Card Security Settings



- 3 On the **Card instance batch load** property sheet, select the XML file containing the card instance definitions, then upload the file to the server.

Click for details.

Viewing Card Contents

To view the contents of a card instance in the Card Manager database:

- 1 Use the Card search facility (see “Figure 31” on page 71) to search for a card or cards. The SIM Card List window is displayed, for example:

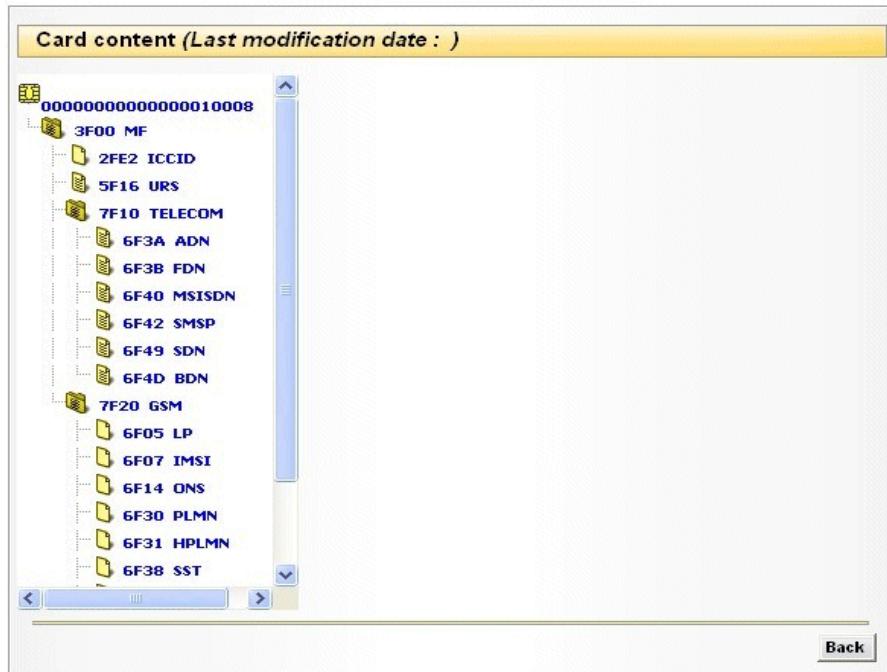
Figure 33 - The SIM Card List Window

SIM card list	
Card MSISDN	Card information
060010000	Serial number 000000000000000010000
060010001	Card profile GX_8K
060010002	Card state Active
060010003	IMSI number 10000
060010004	MSISDN number 060010000
060010005	Linked Card
060010006	Free E2PROM size N/A
060010007	Free volatile size N/A
060010008	Free non volatile size N/A
060010009	Creation date
060010010	Last modification date
060010011	
060010012	
060010013	
060010014	
060010015	
060010016	
060010017	
060010018	
060010019	
	Group identifier

At the bottom are buttons for 'Next' (20/200), 'Update', 'Delete', 'Card content...', 'Security details...', and 'Back'.

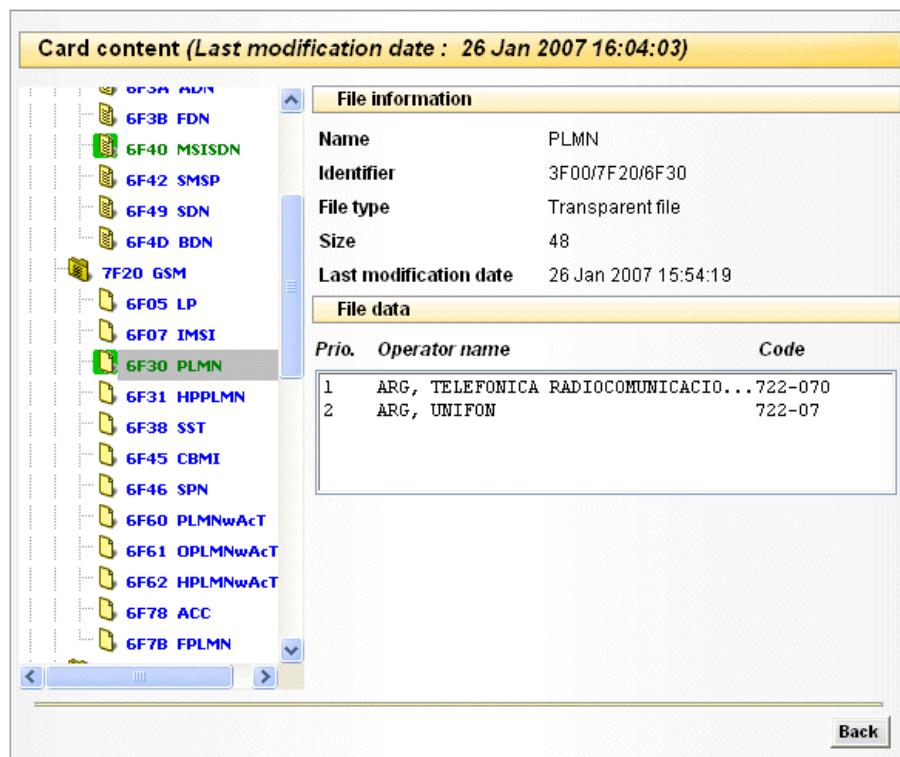
- 2** Select the SIM card in the **Card serial number** column and click **Card content**. The file structure of the card is displayed, as shown in “Figure 34”:

Figure 34 - The Card Content Window



Files for which the content has been modified (that is, that differs from the card profile definition) are highlighted in green, for example,  6F40 MSISDN .

The contents of the card instance depend on the card type. To view the contents of a particular file or application, select an element in the structured view. For example, to view the contents of a file, select an EF file icon in the structured view:

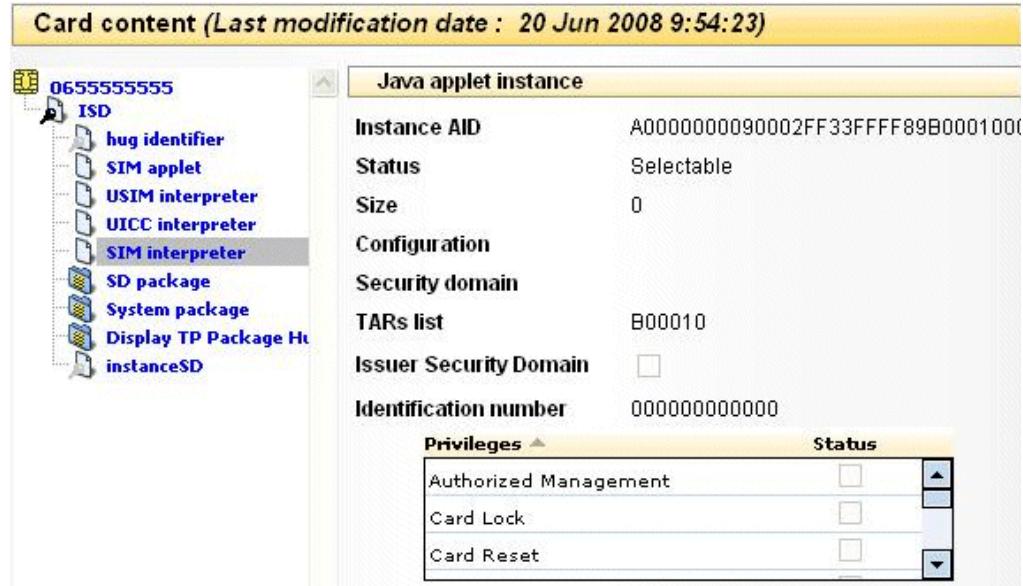
Figure 35 - Properties of a File

You can also display the properties of applications. For example, select a Java applet instance icon in the structured view to display its properties:

Figure 36 - Properties of an Applet Instance

Security domain applications are also represented in the structure view. Click a security domain or one of its components to display its properties. For example, “Figure 37” shows the properties of the SIM Interpreter component of the Issuer Security Domain (ISD):

Figure 37 - Properties of a Security Domain Component



Managing Card Security

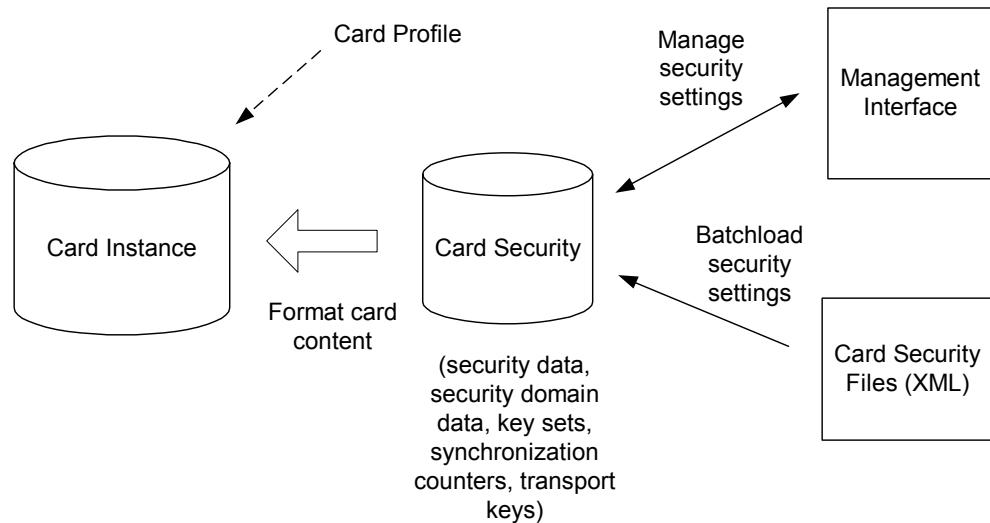
The Card Security Database

The Card Security database stores information about:

- The security settings associated with card instances (key sets, synchronization counters, and so on). These security settings are described in “Additional Security Mechanisms” on page 85.
- Transport keys used to protect batches of cards from unauthorized access during transport of the cards from the card issuer to the telecom operator’s premises. Transport keys are described in “Transport Keys” on page 77.
- Formatting instructions for preparing card-specific messages (formatting classes). Formatting classes are described in the *OTA Manager V5.1 Customization Guide*.

“Figure 38” shows an overview:

Figure 38 - The Card Security Database



Transport Keys

The transport key is an optional security mechanism used to protect a batch of cards' security data (such as security keys and synchronization counters) from unauthorized access during transport from the card vendor to the Telecom operator.

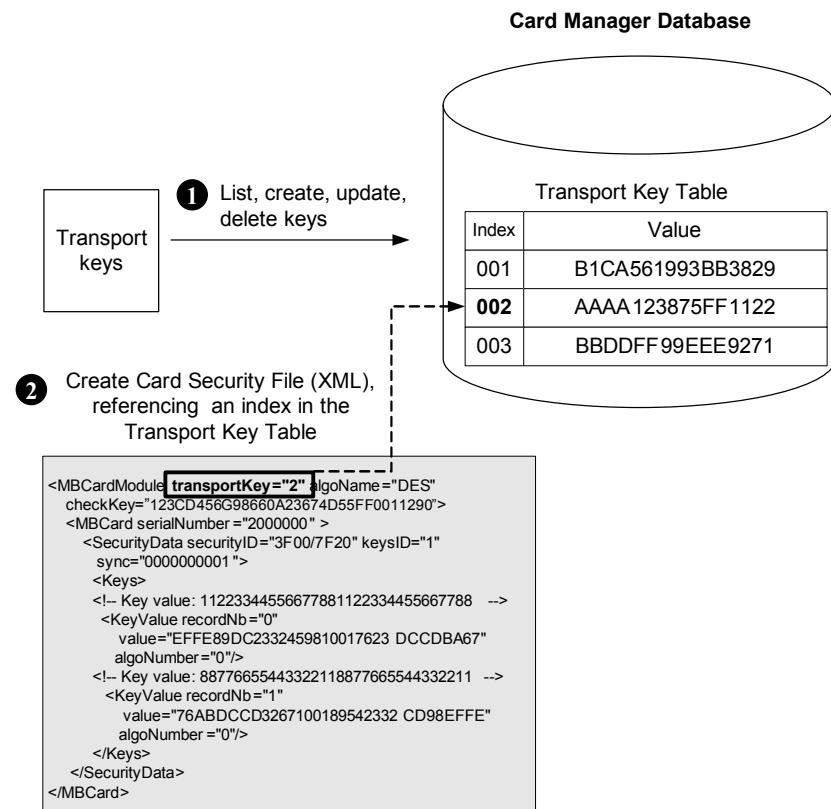
When the card security files are provisioned, data is encrypted using a transport key and algorithm. As an additional security check, the values of these two elements are also used to encrypt a 16-byte hexadecimal shared reference value contained in the **checkKey** attribute.

The card security files only contain the transport key index and the algorithm used to encrypt the security data; the unencrypted value of the transport key never appears in the card security files, and is not even accessible nor visible in the Card Manager management interface windows.

When the transport keys are checked, if the **checkKey** attribute is defined, the XML parsing procedure uses the transport key index and the algorithm to regenerate the hexadecimal **checkKey** value. If the result is different from the shared reference value provided, an error is generated and the provisioning is halted.

The reference (default) value of the **checkKey** attribute is “00112233445566778899AABBCCDDEEFF”

“Figure ” shows an overview of the transport key mechanism:

Figure 39 - The Transport Key Mechanism

Creating Transport Keys

The Card Manager allows you to manage the transport keys stored in the Card Manager database.

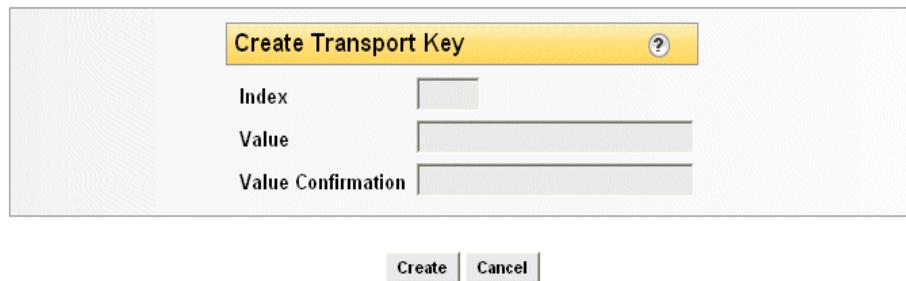
Refer to the online help for information on listing existing transport keys, modifying, or deleting transport keys. Before updating or deleting existing transport keys or indexes, you are required to enter the current transport key that is associated with the index number in question. Note, however, that you can only view transport key indexes, never the actual key values. For security reasons, transport keys cannot be batchloaded.

To create a transport key entry in the Transport Key table of the Card Manager database:

- 1 Do either of the following:
 - On the **Welcome** window, click **Platform management**
 - Click **Platform** on any OTA Manager V5.1 management interface window
- 2 On the left frame of the Platform management window, click **Product configuration**. The Product list window is displayed.
- 3 Select the **Card Manager** card product in the list, and click **Configure**.
- 4 Click **Product interface**.

Figure 40 - The Card Manager Interface

- 5 In the Message Builder Management area, click **Manage** on the **Transport Keys** line. On the Transport Key Index List, click **Create**. The Create Key window is displayed:

Figure 41 - The Create Transport Key Window

- 6 In the window, enter values for parameters shown in "Table 5" and click **Create**.

Table 5 - Transport Key Parameters

Parameter Name	Description
Index	Index number of the transport key between 1 and 999 inclusive. The product rejects any other value and an error is generated.
Value	The transport key value, expressed as an uppercase hexadecimal value.
Value Confirmation	Re-entry of the hexadecimal value entered in the Value field.

Specifying Transport Keys in Card Security Files

Card security definitions must conform with the rules defined in the `messageBuilder.dtd` file.

The following example of a Card Security file illustrates the use of transport keys to create security data for a card instance.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE MBCardModule SYSTEM "messageBuilder.dtd">
<MBCardModule transportKey="6" algoName="TripleDES">
    <MBCard serialNumber="121212" transportKey="5"
        algoName="TripleDES_ECB">
        <SecurityDomain securityDomainID="AID01"
            defaultSync="0000000000" transportKey="4"
            algoName="TripleDES"
            implicitRcAlgoNumber="1" proprietaryRcAlgoNumber="2">
```

The **transportKey** attribute can be specified at various places in the file:

- When the **transportKey** attribute is specified on the **MBCardModule** element, the specified transport key is used for the entire security file.
 - When the **transportKey** attribute is specified on the **MBCard** element, the specified transport key is used for all keys of the card.
 - When the **transportKey** attribute is specified on the **SecurityDomain** or **SecurityData** elements, the specified transport key is used for all keys contained within these elements.
 - When the **transportKey** attribute is specified at the key level (**Kic**, **Kid**, **Kik**, **KeyValue** elements), the specified transport key is used only for that particular key.

Using the **algoName** attribute, a decryption algorithm can optionally be specified (along with a transport key; if none is given, DES is used by default). The following algorithms are supported:

Table 6 - Transport Key Algorithms

algoName	Algorithm
DES	DES in CBC mode
TripleDES	Triple DES in CBC mode
DES_ECB	DES in ECB mode
Triple DES_ECB	Triple DES in ECB mode
XOR	XOR

Batchloading Card Security Settings

Card security settings are batchloaded directly into the Card Security database.

Note: Card security settings can only be batchloaded; there is no management interface equivalent.

The XML structure of the batchload file for security settings differs according to the card type and security model used:

- Native cards with ESMS V1 or ESMS V2
- Java cards with GSM 03.48

OTA Manager V5.1 supports two security models:

- The “Security data” model
- The “Security domain” model

It is recommended to:

- 1 Use the “Security data” model in coordination with Formatting Library V3 service implementations (or any custom formatting library) to provision security settings for native cards (based, for example, on key files). Note that for purposes of backwards compatibility, it is still possible to provision “emulated 03.48” security settings using this model.
- 2 Use the “Security domain” model in coordination with Formatting Library V4 service implementations for provisioning interoperable 03.48 security for Java cards. Note that, for purposes of backwards compatibility, the Formatting Library V4 formatting class is able to handle both the Security Domain and Security Data models (with the same restrictions concerning “emulated 03.48” data as for GemXplore Suite 2.0).

Backwards compatibility is therefore ensured: Formatting Library V3 service implementations still work for cards provisioned with “emulated 03.48” using the “Security data” model, while Formatting Library V4 service implementations still work for cards provisioned with “emulated 03.48” using the “Security data” model.

The Formatting Library V3 formatting class does not support the “Security domain” security model.

XML File Format for Native Cards

To batchload card security settings for a native card, prepare an XML file defining the card security settings. For example:

```

<MBCardModule>
  <MBCard serialNumber="1600920100">
    <SecurityData securityID="3F00/7F20/0001" keysID="1"
      sync="0000000001">
      <Keys>
        <KeyValue recordNb="0"
          value="11223344556677881122334455667788" algoNumber="3"/>
        <KeyValue recordNb="1"
          value="1122334455667788" algoNumber="2"/>
      </Keys>
    </SecurityData>
  </MBCard>

```

Note in particular:

- The **MBCard** element represents a single card instance stored in the database and all associated security parameters that apply to the card. The card instance to which it refers (`serialNumber="1600920100"`) must have been provisioned previously.
- The **SecurityData** element identifies the security component for the card instance, in this case the file `3F00/7F20/0001`.
- The **securityID** attribute is a reference to the key container file on the card. Its value must match that of a security domain card file defined in the card structure area of the card profile. For example:

```

<CardStructure>
  ...
  <CardFile name="GSM" appliNumber="02" type="&DF_FILE;" 
    identifier="7F20" fileSize="15"
    securityDomain="3F00/7F20/0001">
</CardStructure>

```

- The **keysID** attribute is not used for native applications and its value must be set to `"1"`.
- The **sync** attribute gives the synchronization counter value on the card. There is normally only one synchronization counter on a card. It is coded on 5 bytes.
- The **algoNumber** attribute identifies the algorithm to be used. This is an integer value that identifies one of the algorithms supported by the Library V3 formatting library:

Table 7 - Formatting Library V3 Algorithms

algoNumber	Algorithm	Value	Remarks
0	XOR8	8	CC
1	COMP128	16	CC
2	DES CBC	8	CC, Ciphering
3	Triple-DES CBC	16/24	CC, Ciphering
4	COMP128V2	16	CC
5	CRC-32		RC
6	XOR4		RC

- The **recordNb** attribute identifies the key number within the key file. It must match the corresponding certification key reference for ESMS V1 and ESMS V2.

- The **value** attribute represents a key value. Note that the key values may themselves be encrypted with a transport key.

XML File Format for Java Cards (Single Component Keys)

To batchload card security settings for a Java card using single component keys, prepare an XML file defining the card security settings. For example:

```
<MBCardModule>
  <MBCard serialNumber="3000000">
    <SecurityDomain securityDomainID="A000000018434D"
      defaultSyncID="default"
      implicitRcAlgoNumber="6"
      proprietaryRcAlgoNumber="6">
      <Sync value="0000000000"/>
      <Keyset versionNumber="2">
        <Kic value="3042304230443044304530463046"
          algoNumber="12"/> <!-- 3DES/CBC -->
        <Kid value="0123456789ABCDEF100276FEDCBA0123"
          algoNumber="3"/><!-- 3DES -->
        <Kik value="112233445566778899AABBCCDDEEFF00"
          algoNumber="3"/><!-- 3DES -->
      </Keyset>
    </SecurityDomain>
  </MBCard>
  <MBCard serialNumber="3000001">
    <SecurityDomain
      securityDomainID="A000000018434D"
      defaultSyncID="default"
      implicitRcAlgoNumber="6"
      proprietaryRcAlgoNumber="6">
      <Sync value="0000000000"/>
      <Keyset versionNumber="2">
        <Kic value="3042304230443044304530463046"
          algoNumber="12"/> <!-- 3DES/CBC -->
        <Kid value="0123456789ABCDEF100276FEDCBA0123"
          algoNumber="3"/><!-- 3DES -->
        <Kik value="112233445566778899AABBCCDDEEFF00"
          algoNumber="3"/><!-- 3DES -->
      </Keyset>
    </SecurityDomain>
  </MBCard>
```

In this example, two key sets are managed by the same security domain. A default synchronization counter is specified for the security domain, which is used when a message contains only the synchronization counter with no certificate or ciphering.

Note in particular:

- The optional **defaultSyncID** attribute identifies a default synchronization counter to use.
- The **implicitRcAlgoNumber** and **proprietaryRcAlgoNumber** attributes specify the algorithms to use for a redundancy checksum with a SPI set respectively to “Algorithm known implicitly by both entities” or “Proprietary implementations”.
- The **Sync** element contains the value of the keyset’s synchronization counter.
- The **Kic**, **Kid**, and **Kik** elements contain the values of the keyset’s KiC, KID, and KiK keys, respectively. At least one of the **Kic** and **Kid** elements must be included.

- The key length specified in the **value** attribute depends on the algorithm used, and is checked for compatibility with the algorithm during batchloading.
- The **algoNumber** attribute is an integer that identifies the algorithm to use. The value corresponds to one of the algorithms shown in “Table 8 - Security Algorithms” on page 85. The length of the key specified in the **value** attribute must match the key length expected by the formatting library, otherwise an error is returned:

XML File Format for Java Cards (Multiple-Component Keys)

To batchload card security settings for a Java card using multiple-component keys, prepare an XML file defining the card security settings. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE MBCardModule SYSTEM "messageBuilder.dtd">
<MBCardModule>
<MBCard serialNumber="3000000">
<SecurityDomain securityDomainID="A000000018434DFF33FFFF89000000"
    defaultSyncID="default"
    implicitRcAlgoNumber="6"
    proprietaryRcAlgoNumber="6">
    <Sync value="000000001"/>
    <Keyset versionNumber="70" syncID="ISDKeySet70">
        <Sync value="000000001"/>
        <!-- RSA public key -->
        <Key index="1">
            <KeyComponent value="00010001" type="160"/>
            <KeyComponent
value="000086362B4E3B1E20426F8C8557223F4E2B21557F5698F93C963844245B2A5
F44DE335D8C404246475060405D4226E85D35262E66222C6A4D5649EE4B2D62214F3C5
8403C5060902B32FA2134E6546152443156475D3C4C63463C35F0253D313163E3A7CCC
428233B3E355326311E3C465F5CF3471F72401B5B585933916303" type="161"/>
            </Key>
            <!-- RSA private key -->
            <Key index="2">
                <KeyComponent
value="000086362B4E3B1E20426F8C8557223F4E2B21557F5698F93C963844245B2A5
F44DE335D8C404246475060405D4226E85D35262E66222C6A4D5649EE4B2D62214F3C5
8403C5060902B32FA2134E6546152443156475D3C4C63463C35F0253D313163E3A7CCC
428233B3E355326311E3C465F5CF3471F72401B5B585933916303" type="162"/>
                <KeyComponent
value="000000853833C81B51B8391CC829301C443550574C48333694273B2C543F3B4
A6B26B8412E381F29CA5BDA547458415E1A31473F33374A3E3B8CEB5858541C73D7238
85245349DB5222B30A9E9C034243E4D59465E765435544A7F623E63CB402840573724C
624BFF529224F3F453A2E3EF1314B36372E1ADC523563E0375703" type="163"/>
            </Key>
        </Keyset>
    </SecurityDomain>
</MBCard>
</MBCardModule>
```

In this example, one keyset with RSA keys is defined for a security domain:

- The optional **defaultSyncID** attribute identifies a default synchronization counter to use.
- The **implicitRcAlgoNumber** and **proprietaryRcAlgoNumber** attributes specify the algorithms to use for a redundancy checksum with a SPI set respectively to “Algorithm known implicitly by both entities” or “Proprietary implementations”.

- The **Sync** element contains the value of the keyset's synchronization counter.
- The **Key** element contains the definition of a multiple-component key, with its index. Index values 1, 2, and 3 correspond to KiC, KiD, and KiK, respectively.
- The **KeyElement** element contains the value of one component for a key and its corresponding type (also known as algoNumber), as defined in “Table 8 - Security Algorithms” on page 85 and “Table 9 - Additional Algorithms” on page 85.

Security Mechanisms

“Table 8” lists the security algorithms supplied with OTA Manager V5.1.

Table 8 - Security Algorithms

algoNumber	Algorithm	Value	Remarks
0	XOR8	8	CC
1	COMP128	16	CC
2	DES CBC/ECB (use mode requested in KiC byte)	8	CC, Ciphering
3	3-DES CBC	16/24	CC, Ciphering
4	COMP128V2	16	CC
5	CRC-32		RC
6	XOR4		RC

Additional Security Mechanisms

OTA Manager V5.1 can be customized to support additional security algorithms. If the **algoNumber** attribute value is greater than the highest number shown in “Table 9 - Additional Algorithms”, OTA Manager V5.1 searches the `algo.prop` file (in `<OTA-HOME>/RCA/PlugIn/services(mb/format/sms/generic0348)`) for the algorithm number, together with the hexadecimal presentation of the algorithm’s initialization vector, if required.

“Table 9” lists the additional security algorithms. Algorithms with a value greater than 128 are defined in the GlobalPlatform specification.

Table 9 - Additional Algorithms

algoNumber	Algorithm	Value	Remarks
10	DES/CBC/None	8	Ciphering
11	DES/ECB/None	8	Ciphering
12	3-DES/CBC/None	16/24	Ciphering
13	3-DES/ECB/None	16/24	Ciphering
30	CRC-16 A		RC
128	DES/CBC/NOPADDING	8	
129	3-DES/CBC/NOPADDING	16/24	
130	3-DES/CBC/NOPADDING	16/24	
131	DES/ECB/NOPADDING	8	
132	DES/ECB/NOPADDING	8	

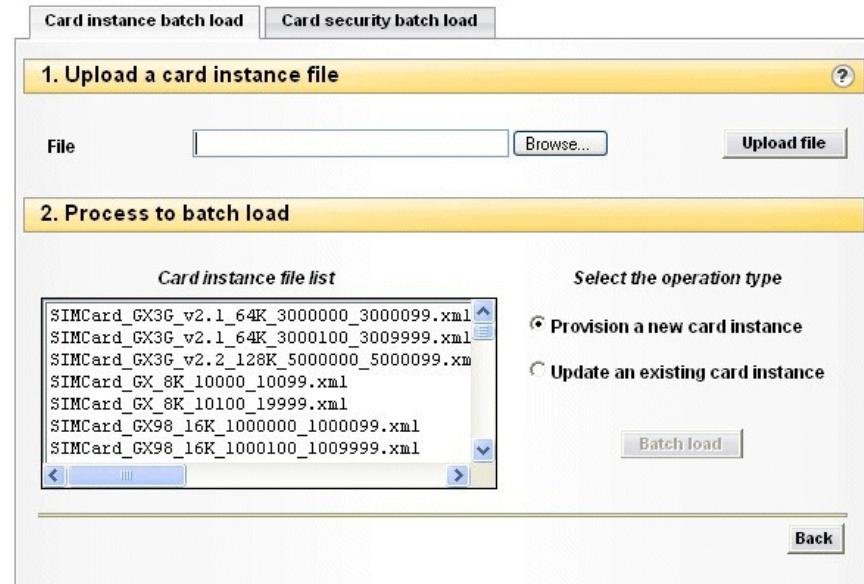
Table 9 - Additional Algorithms (continued)

algoNumber	Algorithm	Value	Remarks
160	RSA	512/1024	Public exponent e component (clear text)
161	RSA	512/1024	Modulus n component (clear text)
162	RSA	512/1024	Modulus n component
163	RSA	512/1024	Private exponent d component
164	RSA	512/1024	Chinese remainder p component
165	RSA	512/1024	Chinese remainder q component
166	RSA	512/1024	Chinese remainder pq component
167	RSA	512/1024	Chinese remainder dp1component
168	RSA	512/1024	Chinese remainder dq1component

Batchloading Card Security Files

To batchload the security files to the Card Security database:

- 1 On the **Search Card Content** window (see “Figure 31” on page 71), click **Batchload**. The following window is displayed:

Figure 42 - Batchloading Card Security Settings

- 2 Select the **Card security batchload** property sheet.
- 3 Click for details.

Viewing Security Settings

To define security settings for the cards listed on the SIM Card List window, click **Security details**.

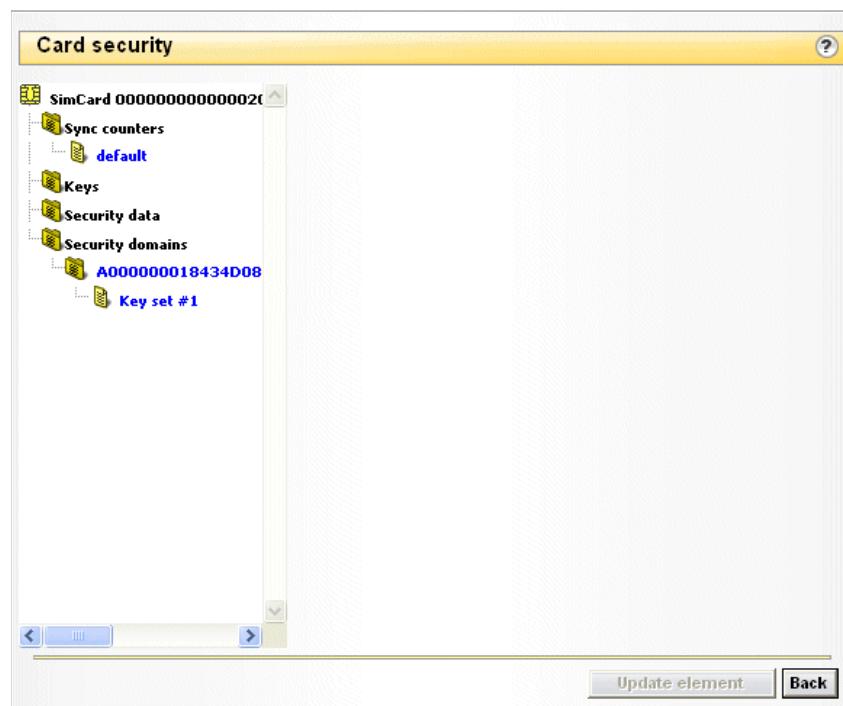
OTA Manager V5.1 supports two security models:

- The Security Domain security model for Java cards, identified by a Security Domain element in the window that appears.
- The Security Data (“native or emulated 03.48”) security model for native cards, identified by a “Security Data” element in the window that appears.

See “Additional Security Mechanisms” on page 85 and “Backward Compatibility of Security Models” on page 286 for more details.

The Card security window shows the current security settings for the card instance:

Figure 43 - Card Security Settings Window



The left-hand panel of the window displays security settings for both “native/emulated 03.48” and “Security Domain” security models, in a hierarchical tree:



The root of the card instance’s security settings. The serial number of the card is displayed alongside the node.



Sub-elements list all the synchronization counters defined on the card.



A Security Data security domain. Can contain either a path to a file on the card (for example, “3F00/7F10/0002”), or the AID of a “native or emulated 03.48” security domain (for example, “A0000000180308”)

**Security domain**

A Security Domain security model. More than one of these elements may be present. Sub-elements of a security domain node can include:

- One or more key sets.
- A default synchronization counter, used for security operations if no synchronization counter is defined for a key set.
- A default algorithm used for RC operations (if the KID security setting specified in the service request specifies use of the “known by both entities” algorithm).
- A proprietary algorithm for RC operations (if the KID security setting specified in the service request specifies use of the “proprietary implementation” algorithm).

**Keys**

The set of all key sets defined on the card. Sub-elements list individual key set files.



For synchronization counters and security domains, each icon of this type identifies the AID of the applet charged with managing the settings. For key sets, it identifies a key set number.

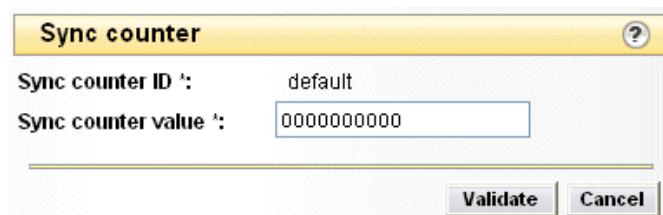
For example:



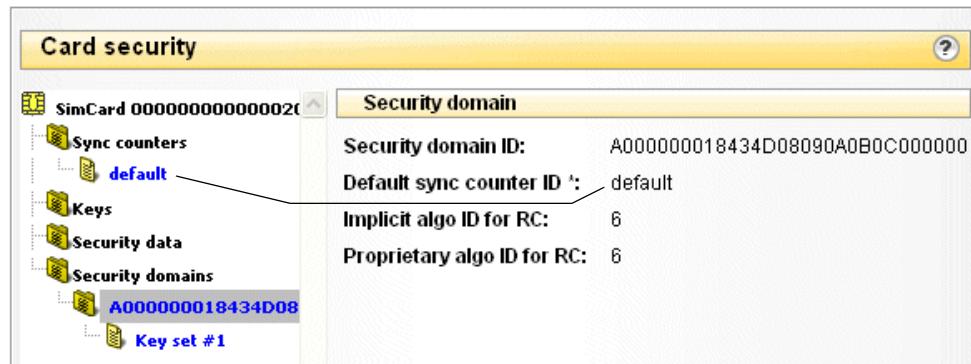
A security domain for a Java card, identified by the AID “A000000018434D08”. One key set is managed by this security domain.

To display or update the properties of any node, select an element in the structure view and click **Update Element**. For example, provided you have the “Card editing” access right selected in your user profile, selecting a Security Data element and clicking **Update element** displays the following window:

Figure 44 - Security Data Window



This window allows you to modify the synchronization counter identifier and key set identifier to use for this card instance. For example, selecting the security data element displays the following pointers:



Click for further details on managing the individual card security settings.

Submitting Service Requests

This chapter describes how to send service requests to target SIM cards.

Note: You can submit requests only when access to the **Request Management** function has been specified in your user profile.

Requests are usually sent “over the air” (OTA). If the subscriber’s mobile phone is switched on, the SIM card is updated immediately the request is sent (within a normal delay, depending on system use, network traffic and so on). If the phone is not switched on, the SMSC continues trying to send the request until the mobile phone is switched on and receives it or until the request times out.

If you have a Wired Internet Reader (WIR) connected to your terminal, you can insert the card in the WIR to modify it. If your PC is not yet set up for this, first see the *OTA Manager V5.1 Installation Guide*.

Each service request has three parts:

- A request destination, where you specify the SIM card to send the request to
- The type of service and the details of the service
- Service request options, such as whether the request should be sent immediately or not.

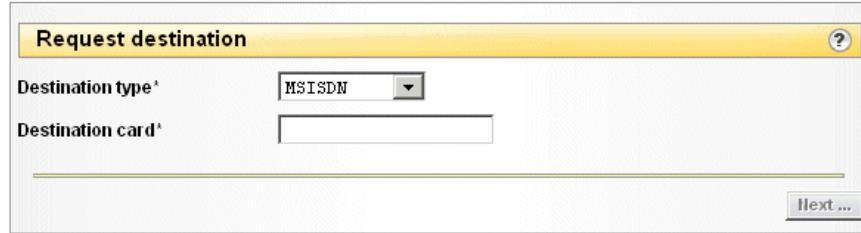
To send a request to multiple cards at once, use a campaign.

Submitting a Service Request

To submit a service request:

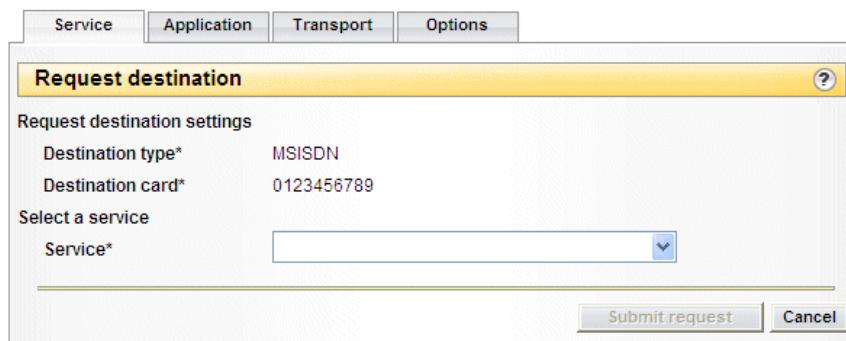
- 1 Do either:
 - On the **Welcome** window, click **Card manager**
 - or:
 - Click **Products** on any management interface window and choose **Card manager**.

- 2 On the left-hand menu, click **Request > Submission**. The **Request destination** window is displayed:



- 3 Choose **MSISDN** as the **Destination type**, then enter the target card's unique MSISDN, for example "0123456789". Click **Next**.

Figure 45 - Request Destination Window

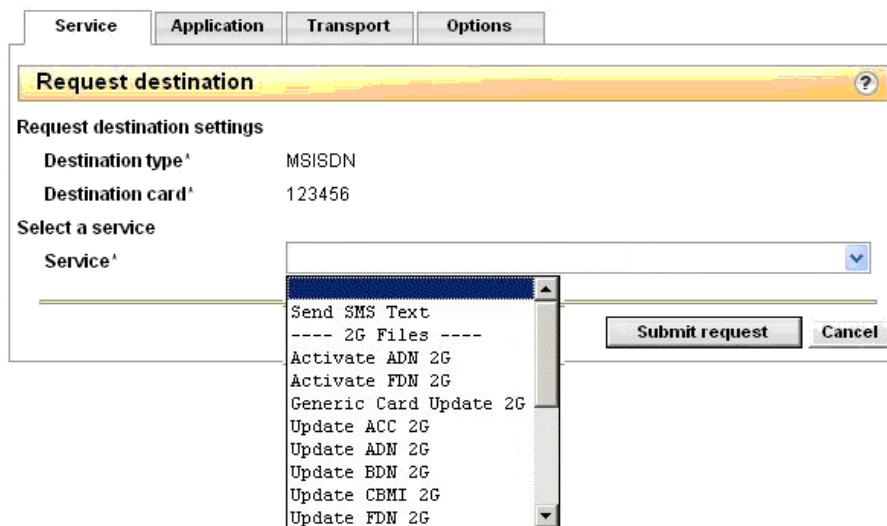


This window has a number of property sheets:

- **Service**. Identify the target card and select the service request to submit from a list.
- **Application**. Allows you to identify the application (for example, applet instance), to target on the card and the security level required by the application.
- **Transport**. Allows you to specify the transport bearer such as SMS, CAT-TP or WIR. In most cases, however, you will use the SMS bearer and the default SMSC, so you do not need to modify these options.
- **Options**. To modify any of the default service parameters.

You can fill in the details on these property sheets in any order before submitting the request. Click for details on individual fields.

- 4 From the **Service** tab, select a service from the drop-down list:

Figure 46 - Service Drop-down List

If the **Destination card** is provisioned in the database, the names that appear in the **Service** list of available services are those that have been allocated to the user in their user profile and that are assigned to the card profile of the selected card. OTA Manager automatically selects the appropriate service implementation of the card profile used by the target subscriber's SIM card.

If the **Destination card** does not exist in the database, the names that appear in the **Service** list of available services are those that have been allocated to the user in their user profile, and that are independent from card profiles. When you select a service, a window with information related to that service is displayed below the **Select a Service** window. Fill in the information required for the service. Click for details on individual fields.

- 5 After providing all details for the service request, click the **Submit request** to send the request. Clicking **Cancel** erases the request details entered and returns you to the **Request destination** window.

If you sent the request over the air, you should see the following:

Request submitted to OTA Manager

This means the request is now being processed by OTA Manager. You can check the progress of the request using the Request Monitoring function as described in "Monitoring Requests" on page 94.

If, for any reason, you want to stop the request before it is processed, you can delete it as described in "The Request Management Window" on page 98.

If you are using a WIR, the physical card's content is updated immediately.

Using the WIR

Note: You can use the WIR only when the **WIR** checkbox has been selected in your user profile.

If you choose the WIR option, you may see a **Security Warning** window when you click **Submit request** to send the request. This is a standard security message that you must confirm in order to use the WIR facility with OTA Manager. If you click

Always trust content from this publisher, you will never see this message again. Otherwise, you will see this message each time you launch a new Explorer window and use the WIR.

Proceed as follows:

1 If prompted to insert the card in the WIR, do it now. (If the card was already inserted, you do not see a message.)

2 Enter the PIN code for the card when prompted and press Enter.

If the card is successfully changed, you can remove the card when the following message is displayed:



If you encounter difficulties using the WIR, refer to “Problems with the Wired Internet Reader (WIR)” on page 270.

Monitoring Requests

OTA Manager allows you to monitor all requests that have been submitted by all users of OTA Manager. You can display all submitted requests, or a subset of requests that you have filtered in order to check on a particular request type or the status of requests, such as, processing, completed, or failed.

Note: You can submit requests only when access to the **Request Management** function has been specified in your user profile. Customer care agents and subscribers can only monitor their own requests. Administrators can monitor all requests, including those of other users.

Request Life Cycle

Once submitted, a request normally goes through the following life cycle stages:

- *Created*: The request has been sent and is waiting to be processed by OTA Manager V5.1.
- *In Process*: OTA Manager V5.1 is processing the request, or has processed the request and has sent it to the operator's network gateway (such as the SMSC for SMS messages, or the GGSN for CAT-TP) to be sent to the subscriber's mobile. It is waiting for a reply.

The following states are also assigned if the request was submitted with a guarantee of delivery:

- *Succeeded*: All messages have been successfully sent to the SIM card.
- *Failed*: A message has been returned indicating that the request has failed.

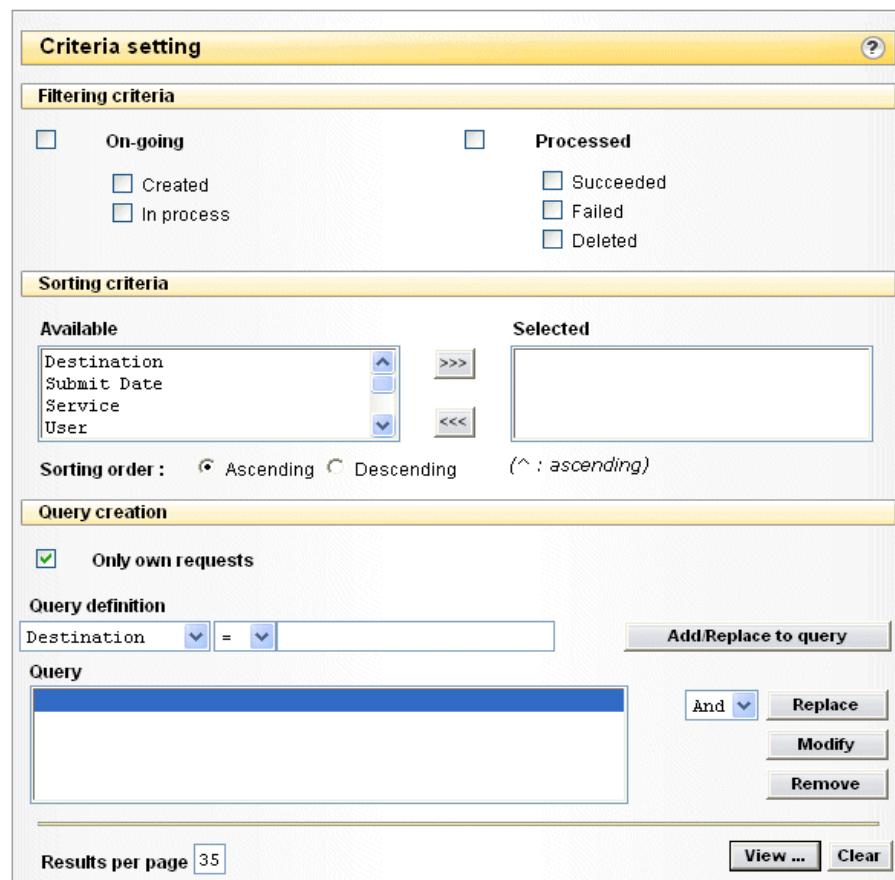
- **Deleted:** The request was stopped by the customer care agent before it could be processed.
- **Expired:** The request failed because the **Validity Date** limit was reached before it could be completed.
- **Suppressed:** You stopped the request after it is sent by OTA Manager V5.1, but before it is sent from the SMSC to the mobile phone. This status only applies to requests stopped by an administrator.

Searching For Requests

To monitor service requests:

- 1 Do either of the following:
 - On the **Welcome** window, click **Card manager**
 - Click **Product** on any management interface window and choose **Card manager**.
- 2 On the left hand menu, select **Request > Monitoring**. The Criteria setting window is displayed:

Figure 47 - Criteria Setting Window



In this window, you can define exactly what types of request you want to check. If you want to check *all* requests, click **View** at the bottom of the window without entering anything else. Otherwise, fill in the parts of the window to narrow down your search. View help for details.

Defining Queries

If you do not enter any queries in the **Criteria setting** window, the report lists all requests submitted that have not yet been acknowledged or purged from the database. There may be a great number of these, so it is more useful to try and list only the requests that you are interested in. You do this by entering *queries* in the bottom part of the **Criteria setting** window.

Only queries with the states CREATED, IN PROCESS, FAILED, or EXPIRED can be searched for and monitored. Queries with the state SUCCEEDED can only be monitored if the user selected the **Request acknowledgement** option when submitting the request.

To build a query, you define each expression first on the **Query definition** line and then move it to the **Query** area:

- 1 Optionally, select the **Only own requests** option. This limits the search to requests that you yourself initiated. Deselect this option to search for all users' requests, including your own.

Note: This option is only displayed if you are currently logged in as an administrator. It is not available if you are logged in as a customer care agent or subscriber—these types of user can only search for requests that they themselves initiate.

- 2 On the **Query definition** line:
 - a) Select a field from the first drop-down list.
 - b) Select a comparison operator: equals (=), is not equal to (!=), is less than (<), or is greater than (>)
 - c) Enter a text value in the text box according to the following formats:

Table 10 - Service Query Parameters

Request attribute	Format
Request ID	Exactly as specified (see "Figure 48")
Service	Must be exactly as spelled in the service drop-down list, though spaces and upper/lowercase are ignored. For example, the following are all valid: Update LP UpdateLP updatelp
Destination	Can be an MSISDN, IMSI, or ICCID. The format must be exact, including spaces. (The MSISDN is as defined when provisioning subscribers.)
Submit Date	Click the calendar icon. Adjust the time if necessary and then select a date. To specify a particular day you would enter two expressions like this: Submit Date > 17-Apr-2002 8:00:00 AND Submit Date < 17-Apr-2002 18:00:00

Your Expression Definition may now look something like this:

The screenshot shows the 'Criteria setting' window with the 'Query definition' and 'Query' sections visible. In the 'Query definition' section, there is a dropdown menu set to 'Submit date', a comparison operator dropdown set to '<', and a date/time input field containing '26 Jan 2007 12:00:00'. To the right of these fields are 'Add' and 'Replace to query' buttons. In the 'Query' section, there is a large text input field, an 'And' dropdown menu, and three buttons: 'Replace', 'Modify', and 'Delete'.

3 Click Add/Replace to query.

The expression is copied to the **Query** area. For example:

The screenshot shows a 'Query' window with a text input field containing the expression: 'Submit Date < 26 Jan 2007 12:00:00'. To the right of the input field are three buttons: 'And' (highlighted in blue), 'Replace', and 'Modify'.

This will list all requests submitted before 26th January 2007.

You can combine several expressions in one query to further narrow the filter. For example:

The screenshot shows a 'Query' window with a text input field containing the expression: 'Service = Update PLMN 2G AND Submit Date < 26 Jan 2007 12:00:00 OR Destination = 06 78 56 34 23'. To the right of the input field are three buttons: 'Or' (highlighted in blue), 'Replace', 'Modify', and 'Remove'.

A request must satisfy all three of these expressions for it to be included in the list, because the expressions are combined using the word AND.

Alternatively, you can combine expressions using the word OR:

The screenshot shows a 'Query' window with a text input field containing the expression: 'Service = Update PLMN 2G AND Submit Date < 26 Jan 2007 12:00:00 OR Destination = 06 78 56 34 23'. To the right of the input field are three buttons: 'Or' (highlighted in blue), 'Replace', 'Modify', and 'Remove'.

This means: list all requests that were submitted before 26th January 2007 12h00 with service request **Update PLMN 2G** or requests to the mobile phone 06 78 56 34 23 (MSISDN). That is, the list may include two different groups of requests.

You can build as many expressions as you want into your query. If you use both AND and OR operators, the program connects the AND expressions first and then separates the groups of expressions using the OR operators. For example (where Exp is a complete expression):

Exp1 And
Exp2 And
Exp3 Or
Exp4 And
Exp5

is interpreted as:

(Exp1 And Exp2 And Exp3) Or (Exp4 And Exp5)

So there are two groups of requests searched for in this expression.

4 Click View at the bottom of the window to run the search.

Note: To remove a line in the query, select it and click **Remove**.

To modify a line in the query, select it, click **Modify**, make changes to it on the **Expression formulation** line, then click **Add expression**. It replaces the original line in the query

To change the AND/OR operator for a line, select the line, click the down arrow and select the operator, and click **Add**.

The Request Management Window

When you click **View** in the **Criteria setting** window, OTA Manager V5.1 searches through the Framework's Invocation registry database for all requests that satisfy the criteria you entered. It then lists these in a viewing window like the one shown here, sorted as you specified. You may need to scroll to see all the requests in the window.

Figure 48 - Request Management Window

Request management					
Submit time zone : Central European Time					
Submit date	User	Transaction ID	Service	Destination	State
<input type="checkbox"/> 26 Jan 2007 16:42:20	admngemplus		Update ADN 3G	063100055	MISSING_CARD*
<input type="checkbox"/> 26 Jan 2007 16:04:03	admngemplus		Update MSISD...	0603000055	SUCCEEDED*
<input type="checkbox"/> 26 Jan 2007 15:57:50	admngemplus		Update MSISD...	0603000055	SUCCEEDED*
<input type="checkbox"/> 26 Jan 2007 15:54:19	admngemplus		Update PLMN 2G	0603000055	SUCCEEDED*
<input type="checkbox"/> 26 Jan 2007 15:49:39	admngemplus		Update PLMN 2G	0603000000	SUCCEEDED*
<input type="checkbox"/> 26 Jan 2007 12:10:57	admngemplus		Update PLMN 2G	0602000055	FAILED*
<input type="checkbox"/> 25 Jan 2007 17:43:13	admngemplus		Update PLMN 2G	0602000520	MISSING_CARD*

* : the invocation is waiting to be acknowledged by the sender

7/7 [Previous](#) [Next](#) [Refresh](#) [Delete](#) [Ack](#) [Replay](#) [Select all](#) [Back](#)

However, the list is limited in the following ways:

- It lists all submitted requests, unless the **Only own requests** check box is selected on the Criteria Setting window (see "Figure 47" on page 95). In this case, only those requests submitted by you are listed.
- Requests awaiting acknowledgement are shown with an asterisk (*) next to them
- Requests being processed are shown as "IN_PROCESS", but once they have succeeded they are not listed unless a guarantee of delivery was set for the request. Failed requests remain listed.
- Acknowledged requests disappear from the list and are removed from the database.
- Each request is only listed for a limited period of time. Once this time limit is reached, it is automatically removed ("purged") from the database in order to prevent the database growing too large with obsolete records. To set up the purge time, see "Backing Up Data" on page 19.

If the requests do not all fit on one page, you can use the **Next Block** button to see other pages of requests.

Note: The window shows a snapshot of the state of the requests; it does not update itself automatically as the request states change. However, you can update it at any moment by clicking **Refresh**.

The window fields are explained in the online help.

Suppressing a Request

If requested by a Customer Care Agent, you can attempt to suppress a request if it has already left OTA Manager V5.1. This can be checked by viewing the request details. Click its  icon; if the State is IN_PROCESS (SENDING), the request has already been sent to the SMSC. To suppress the request, select it and click **Delete**.

When this attempt is made, the state of the request changes to TRYING_TO_SUPPRESS. If the Suppress operation works, the request's state changes to SUPPRESSED. If the Suppress operation does not work, the state changes back to IN_PROCESS.

Checking the Details of a Request

To see more details of a request, click the  icon. This allows you to see the full text of any error message so you can investigate, for example, why a request failed:

Figure 49 - Request Details Window



Select any entry in the list to view its details in the **Submission result** area of the window.

Click **Card content** to view the contents of the card. See "Figure 34" on page 74 for an example of the card content that is displayed.

If a request has failed, you resubmit the request by clicking **Replay** on the Request Management window (see "Figure 48" on page 98).

Managing Campaigns

This chapter describes the campaign engines available with OTA Manager V5.1, how to choose the most suitable campaign engine for any particular campaign, and how to tune each of the campaign engines.

Choosing a Campaign Type

OTA Manager V5.1 offers two different campaign types—standard campaigns and massive download XCT campaigns.

Both campaign types are accessed via the Campaign Manager. The Campaign Manager is a high-level module that allows you to create campaigns that work with other products such as Service Manager and Device Manager.

XCT campaigns are offered via an additional module integrated within the Campaign Manager product.

The XCT module has been designed to run very fast campaigns targeted at a very large number of cards. In order to achieve this level of optimization, the XCT module has a number of constraints that prevents it running every type of campaign update. It does, however, cover the most common bulk update situations.

It is therefore important to understand the differences between standard Campaign Manager campaigns and XCT campaigns in order to be able to choose the most suitable campaign type.

The following describes the criteria you can use to choose which campaign engine to use:

- In general, if your campaign is to target a very large number of subscribers and must be completed in the shortest possible time, use an XCT campaign. Standard campaigns, on the other hand, provide greater functionality and flexibility, so are better adapted to smaller, more personalized campaigns.

Be aware, however, that the performance of an XCT campaign can be negatively affected if a large number of unitary service requests or other campaigns target the same cards as the XCT campaign at the same time.

- Only one XCT campaign can be active or scheduled at any one time, whereas multiple classic campaigns can run concurrently. This means that a new XCT campaign cannot be scheduled while an existing XCT campaign is still running (that is, has not yet attained a “FINAL” state, as shown on the Campaign Monitoring window).

To work around this, you could consider splitting up an XCT campaign into a number of different card sets. For example, you could define a “pre-paid” card set and a “post-paid” card set within the same campaign.

- If your campaign uses services that generate different card content updates, you must use a standard campaign. If you are unsure about a particular service's compatibility, check with your Gemalto representative.
- If you require billing information to be generated for a campaign, you must choose a classic campaign; XCT campaigns cannot generate billing ticket information.
- If your campaign requests and processes PoR messages returned by the target card and the destination address is hard-coded in the subscriber's SIM card, you must use a classic campaign. Destination addresses are not, however, hard-coded in SIM cards that conform to the relevant standards.
- XCT campaigns only work in MT transmission mode; if SMSC or TRANS transmission mode is required, you must use a classic campaign.
- XCT campaigns can be used in conjunction with the SMPP, CMG and NOKIA SMSC channel drivers.

"Table 11" summarizes how to choose a campaign engine:

Table 11 - Choosing a Campaign Type

Criteria	Use a standard campaign?	Use an XCT campaign?
Campaign must run very fast	Not optimal	Strongly recommended
Campaign targets huge number of cards	Not optimal	Strongly recommended
Multiple-services campaign	Possible	Possible
Run several campaigns simultaneously	Mandatory	Not possible
SIM card content check required before sending	Mandatory	Possible, with restrictions. It is not possible to perform a card content check for individual cards.
Call Detail Record (CDR) required for billing	Mandatory	Not possible
Use SMSC or TRANS mode	Mandatory	Impossible (only MT mode is supported)
Use a channel driver other than SMPP, Nokia or CMG SMSC	Mandatory	Impossible (only SMPP, Nokia and CMG SMSC are supported)

Setting Up and Running a Campaign

Once you have chosen the most suitable campaign engine, the basic steps involved in setting up and running a campaign are similar for both campaign types:

- 1 Creating a *scenario*. A scenario is the service or sequence of services to be executed, in an order that you define. At this point you also define the type of campaign—standard or XCT—that you want to run.
- 2 Defining the targets of the campaign. Targets are the SIM cards of the subscribers that the campaign is aimed at. For ease of management, the targets can be defined in a target list; an XML file that you create then upload to the server.
- 3 Once the scenario and targets have been defined, scheduling and submitting the campaign.
- 4 Monitoring the progress of the campaign.

- 5 If necessary, resubmitting the campaign to those SIM cards that were not successfully updated the first time for any reason (for example, the subscriber's handset was switched off at the time).

Note: With standard campaigns, if a default SMS channel driver is chosen during campaign scheduling, the SMS channel driver velocity set in the profile is no longer taken into account. Consequently, the SMS channel driver configured with LOW velocity is used to send campaign messages.

See “Activating and Deactivating the Audit Trail” on page 32 and “Setting the SMS Channel Driver Velocity” on page 34 for more information on channel drivers and channel driver velocity.

To begin using the Campaign Manager:

- 1 On the top menu, click **Products** and choose **Campaign Manager** (or click the **Campaign Management** button on the Welcome window). The left-hand menu then displays XCT campaign-specific menus. These options are only present if the optional XCT module is installed.
- 2 Click **Help** in the left-hand menu for help specific to campaign tasks.

How the Campaign Manager Works

The Campaign Manager builds a campaign using a list of services grouped together in a campaign scenario, and associating this scenario with a target list. The target list file first has to be uploaded to the platform server.

When the campaign is created, the database tables relevant to the campaign are populated in the database. One of these tables stores the status of the campaign on each target, the other stores the status of each service execution on each target.

A campaign starts when the programmed start time is reached. The life cycle proceeds as follows:

- 1 The Campaign Scheduler searches the database for targets that are ready for execution and elects them. The target election process builds a valid unitary task (one service plus one target) and uploads them to memory.
- 2 The unitary task executions cause invocation submissions on associated product(s), as defined in the campaign scenario.
- 3 Traffic flow between Campaign Manager and the connected product is controlled using the **Maximum Throughput** and **Maximum Congestion** parameters:
 - **Maximum Throughput**, expressed as a number of requests per second, is the rate at which requests are transferred from Campaign Manager to the product. The value of this counter should be set in relation to the rate at which the SMSC and the underlying product can process requests.

This parameter can also have an “infinite” value, meaning that no restrictions are made on the traffic flow between the Campaign Manager and the product.

Note: No restrictions are made by Campaign Manager. This does not mean, however, that the connected product cannot operate a traffic flow control.

If the connected product blocks the traffic, Campaign Manager automatically adapts its submission speed to avoid intensive CPU usage.

-
- **Maximum Congestion** is the absolute maximum number of requests that the product can manage, which prevents it from becoming overloaded with submissions from Campaign Manager. This is useful in limiting the Campaign

Manager against other unitary flows when the business flows are not separated (one product for all use cases).

- 4 An acknowledgement that the unitary task has been executed is returned by the product and Campaign Manager writes the status of the task to the relevant table in the campaign database.

The cycle continues until all the unitary tasks have been executed successfully or a programmed event is raised (for example, campaign end date is reached).

For this system to work efficiently, and a successful campaign run, the Validity Period needs to be set correctly.

Different strategies are possible when creating the campaigns. But it is important to understand the platform's behavior:

- 1 The campaign is created in the database
- 2 All the cards of the campaign are loaded in a specific table for campaigns
- 3 The platform creates jobs
- 4 Each job formats the SMS messages and sends them to the SMSC channel driver
- 5 When a job is terminated, a new job is created with a card from the database.

Tuning XCT Campaigns

The following describes how you can tune the XCT campaign manager to achieve optimum performance.

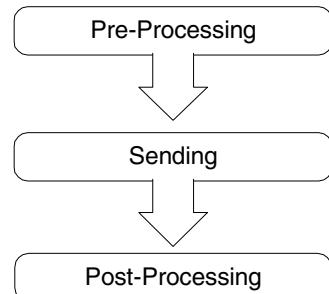
Overview

The XCT campaign manager engine is composed of three separate modules:

- The Pre-Processing Module
- The Sending Module
- The Post-Processing Module

These three modules perform their processing in distinct phases; pre-processing, sending, and post-processing:

Figure 50 - XCT Processing Phases



You can specify processing timeslots in which individual phases start and finish: refer to "Step 2: Defining Campaign Parameters" on page 107 for details.

The Pre-Processing Phase

When you submit a campaign, XCT creates an *invocation* for each subscriber targeted by the campaign.

The Pre-Processing module sends batches of these invocations to OTA Manager V5.1's Card Manager module, which retrieves any necessary service parameters, card content (from the Card database), and security settings (from the Card Security database). The Card Manager then uses the appropriate formatting library to generate a series of formatted SMS messages for the invocation. Each invocation may generate one or more messages—for example, a campaign to download an applet onto a million SIM cards would generate a million separate invocations, each comprising perhaps 30 SMS messages.

Note that when an invocation consists of a sequence of SMS messages, a PoR is only requested in the last command packet.

The Card Manager returns the prepared SMS messages to the pre-processing module, which stores both the invocations and the generated SMS messages in a dedicated XCT database ready for sending.

Note also that, unlike the “classic” campaign engine, XCT stores SMS messages in the XCT database and not in memory. This allows campaigns that are paused or interrupted for whatever reason to be restarted without loss of data, and enables you to monitor each individual invocation in detail (see “Setting the Maximum Number of Retries and the Retry Delay” on page 108). This also has a considerable impact on the parameter values used for a campaign; see “Step 2: Defining Campaign Parameters” on page 107 for more information.

The Sending Phase

When pre-processing is complete, the Sending module begins sending messages.

The Sending module uses dedicated instances of SMS channel drivers to send messages to SMSCs and SS7 routers. The SMSC is the default. To function with SS7 routers, OTA Manager V5.1 must be associated with the High Speed SMS Router product.

Submit responses, Delivery Status responses and PoR codes returned by the SMSC or SS7 routers are processed by the Sending module: all results, including PoRs and mobile-originated messages, are stored in the XCT database.

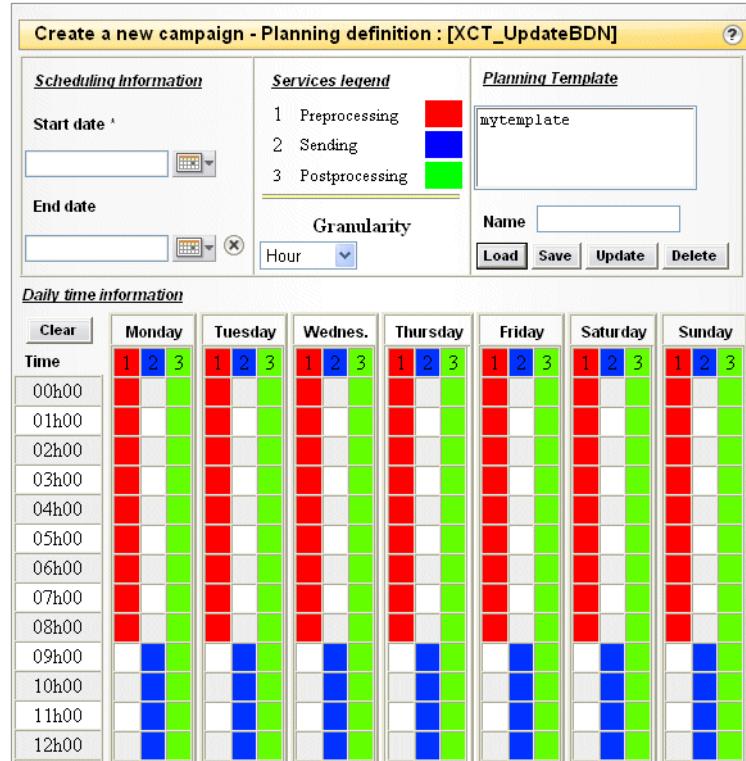
The Post-Processing Phase

The Post-Processing module sends the results of invocations to the Card Manager in batches. The Card Manager then updates the card contents in the Card Manager database to reflect the new contents of the SIM card (synchronization counters, successfully downloaded applets, and so on).

Step 1: Scheduling XCT Campaigns

You define processing timeslots for the three campaign phases on the planning window of the campaign creation wizard:

Figure 51 - Defining Campaign Timeslots



On this window, you specify the start date of the campaign and, optionally, the required end date (if you do not specify an end date, you must stop the campaign manually).

You then define processing timeslots, relative to the campaign start date, by dragging the mouse across the required timeslots. You can save and reuse the current layout and selected timeslots as *planning templates*.

Click for details.

The pre-processing phase is typically the most processing-intensive phase of an XCT campaign, as it is highly dependent on the Card Manager module's performance in formatting SMS messages. It is therefore recommended to schedule pre-processing for when platform usage is likely to be at its lowest, for example, overnight or at weekends.

The sending phase should be scheduled during office hours, as this is when the highest proportion of handsets are likely to be switched on.

During the sending and post-processing phases, the card currently being processed is unavailable for all other OTA updates between the sending of the first SMS to the card and completion of the post-processing phase. Therefore, to minimize the period of unavailability of a card, it is recommended to schedule post-processing for 24 hours a day, from the beginning to the end of the campaign (in general, the post-processing phase is not as processing-intensive as pre-processing or sending).

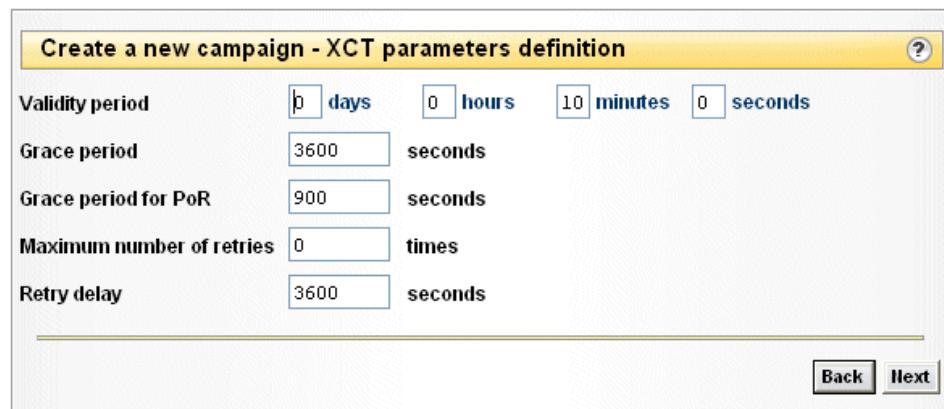
Note: You must ensure that the Card Manager's configuration (card profiles, services, security settings, and so on) is not modified between the pre- and post-processing phases, as this can cause unpredictable results. For this reason, it is recommended not to leave long periods between the two processing phases.

The SENDING_PAUSE_MODE_FINISH_INVOCATIONS product parameter can be used to specify that XCT finishes sending all messages to a particular card before beginning post-processing. The default value is “True”. Refer to “Product Parameters” on page 171 for details.

Step 2: Defining Campaign Parameters

The XCT Parameter Definition window allows you to define XCT campaign parameters:

Figure 52 - XCT Parameters Definition Window



Setting the Validity Period

The **Validity period** specifies the SMSC expiration time, after which an invocation is discarded if it has not been delivered to the destination. This parameter should be configured differently depending on whether XCT is connected to an SMSC or SS7 router.

The validity period setting should also be set in conjunction with the FLOW_CONTROL product parameter setting (see “Appendix B - Product Parameters”).

The value you enter in the **Validity Period** boxes is rounded as follows:

- For values less than 12 hours, to the nearest 5 minutes.
- For values between 12 hours and 1 day, to the nearest 30 minutes.
- For values between 1 day and 30 days, to the nearest 1 day.
- For values between 30 days and 63 weeks, to the nearest 1 week.

All validity period processing is performed during the Sending phase.

SMSC Mode

Because XCT invocations are stored in a database, increasing the validity period does not have an negative impact on system throughput. Therefore, it is recommended to set a relatively long validity period, for example, 3 days. During this period, the SMSC uses its own retry mechanisms to resend any messages that it has failed to deliver.

You should also assign a high value to the FLOW_CONTROL product parameter in this case (recommended value is 3,000,000).

If you set both a long validity period and high FLOW_CONTROL value, a high service execution success rate can be expected.

SS7 Mode

The SS7 router uses an alert-based mechanism to inform XCT whenever a subscriber returns within network coverage. The validity period is used by the OTA Manager V5.1 platform only; the SS7 router itself ignores the validity period parameter value. The recommended validity period value is 10 minutes.

You should also assign a low value to the FLOW_CONTROL product parameter in this case (recommended value is 10,000). The SS7 router itself does not take this parameter value into account.

Setting the Grace Period

The *grace period* is the time that XCT waits for a Delivery Status (DS) response message after the validity period has expired before assigning a final status code to an invocation. The grace period allows you to handle situations where responses do not arrive because of network overload. When the grace period expires, the Sending module sends an Inquiry message to the SMSC (if the SMSC supports it), then assigns a final status code to the invocation depending on the result of the Inquiry.

It is recommended to allocate a grace period value of 1-4 hours for SMSC mode (depending on the validity period defined), and 5 minutes for SS7 mode.

If the current message is the last in a sequence of concatenated messages for which a PoR has been requested, then the Sending module must wait for a PoR response from the SIM card. The **Grace Period for PoR** is the period of time to wait after expiration of the validity period for a PoR to arrive before assigning a final status code to the invocation, depending on the Card Manager SUCCESS_ON_POR_EXPIRED parameter.

It is recommended to allocate a **Grace Period for PoR** value of 10 minutes for both SMSC and SS7 mode.

All grace period processing is performed during the Sending phase.

Setting the Maximum Number of Retries and the Retry Delay

The **Maximum number of retries** is the number of times that XCT resends messages that could not be delivered. This parameter is valid for SS7 mode only; when in SMSC mode, the parameter should always be set to zero (0) to indicate no retries (as the SMSC uses its own retry mechanism).

Note: Unlike classic campaigns, in which all messages of an invocation must be reformatted before being resent, XCT only starts resending from the message that failed. For example, if the fifth message in a concatenation sequence of six fails, only the fifth and sixth messages are resent.

The **Retry delay** is the period of time before resending a failed message. Because retrying messages has a lower priority than sending messages, this is a minimum period of time and is not guaranteed.

This parameter is valid for SS7 mode only. It is recommended to set the **Retry delay** parameter value to 86400 seconds (1 day) and the **Maximum number of retries** parameter to the number of days in the campaign.

Step 3: Determining SMSC Channel Parameters

Configuring the SMSC Channel for XCT Using Product Parameters.

To modify product parameter values:

- 1 On the top menu, click **Products** and choose **Campaign Manager** (or click the **Campaign Management** button on the Welcome window).

- 2 Click **Platform** in the top menu.
- 3 Click **Product config.** on the left-hand menu.
- 4 Select the product instance in the product list and click **Configure**.
- 5 From the Product Configuration window, click **Edit Parameters** to display the product parameters list:

Figure 53 - XCT Product Parameters

Product parameters list		
Group	Name	Value
CHANNEL	ALIVE_PERIOD	0
CHANNEL	INQUIRY_ACTIVATED	false
CHANNEL	LOCAL_PORT	0
CHANNEL	LOGIN_RETRY_PERIOD	10
CHANNEL	LOGIN_TIMEOUT	10
CHANNEL	RETRY_NUMBER	0
CHANNEL	RETRY_TIMEOUT	10
CHANNEL	SMSC_IP_ADDRESS	6011701
CHANNEL	SMSC_IP_PORT	8003
CHANNEL	SMSC_PASSWORD	passwd
CHANNEL	SMSC_TYPE	CMG
CHANNEL	SMSC_USER	NULL
CHANNEL	SS7_MODE	false
CHANNEL	STACK_SIZE	1000

- 6 For help on editing the parameters, click .

Configuring the **STACK_SIZE** Product Parameter

The **STACK_SIZE** product parameter specifies the size of the SMS buffer within the SMSC driver, containing messages ready to be sent. This parameter value should be as small as possible, for the following reasons:

- To limit the number of SMS messages lost in case of an SMSC disconnection.
- To limit the “latency” of the system: if an OTA update is comprised of several SMS messages and the **STACK_SIZE** parameter value is large, then the time to update a single card will be long, since every SMS message must pass through the SMS buffer.

However, there is a minimum size for this buffer because the XCT Sending module “fills” this buffer only once every half a second.

Recommendation: Set the **STACK_SIZE** parameter to the same value as the bandwidth (in terms of SMS per second). For example, if the SMS bandwidth dedicated to XCT is 25 SMS/second, then set **STACK_SIZE** to 25.

Configuring **SS7_MODE**, **INQUIRY_ACTIVATED** and **FLOW_CONTROL** Parameters for SMSC Mode

When XCT is used with an SMSC, it relies on the SMSC’s retry mechanism for SMS delivery. As a consequence, the **PARAMETERS/FLOW_CONTROL** product parameter (the number of SMS messages sent for which a DS has not yet been received) should be set to a large number (3,000,000 by default), for example:

```
CHANNEL, SS7_MODE=false
PARAMETERS, FLOW_CONTROL=3000000
```

Depending whether the SMSC can respond with the SMS status at the end of the Validity Period, the CHANNEL/INQUIRY_ACTIVATED product parameter must be set to either “False” or “True”.

Note that some SMSCs (for example, the CMG SMSC) delete a message when it has been successfully delivered or when it has expired. In this case, there is no point using the XCT Inquiry mechanism, because XCT always performs the inquiry at the end of the Validity Period (+Grace period), so the response from the SMSC will always be “unknown message”, and no additional information is obtained.

Consequently, for the CMG SMSC, always set the CHANNEL/INQUIRY_ACTIVATED product parameter to “False”. In fact, Inquiry frames are not available in the Gemalto implementation of the CMG protocol.

Configuring SS7_MODE, INQUIRY_ACTIVATED and FLOW_CONTROL Parameters for SS7 Mode

When XCT is used with a BMD SS7 router, the maximum number of cards that are targeted in parallel must be limited, due to memory constraints in the SS7 router. The maximum number of targeted cards must be specified by the administrator of the SS7 router and must be configured in the PARAMETERS/FLOW_CONTROL product parameter to ensure that XCT does not overflow the SS7 router, for example:

```
CHANNEL, SS7_MODE=true
CHANNEL, INQUIRY_ACTIVATED=false
PARAMETERS, FLOW_CONTROL=10000
```

Note: The maximum number of cards targeted in parallel must be at least equal to the throughput multiplied by the average time to deliver a SMS. If less than this figure, the maximum throughput will never be reached because XCT will stop sending when the maximum number of cards targeted in parallel is reached.

In order to evaluate the average time to deliver a SMS, perform an SQL request on the database view XCT_VIEW_SMS_STATS.

Step 4: Configuring the SMSC

It is important to tune the SMSC to match the XCT campaign engine’s configuration. For this, you must contact the SMSC’s operator.

Points to consider include:

- 1 Configure the SMSC so that it has sufficient temporary storage to match the values of the Validity Period and FLOW_CONTROL product parameters set on the OTA Manager V5.1 platform. For example, if the FLOW_CONTROL product parameter is equal to 3,000,000, the SMSC must be configured with sufficient memory and buffer space to store at least this number of messages.
- 2 The retry mechanism on the SMSC side must be configured to match the FLOW_CONTROL parameter value set on the XCT side. For example, if the FLOW_CONTROL value is 3,000,000 and the retry value is set to 3 minutes, overflow is likely to occur. A typical value to set is 1 retry per day.
- 3 The SMSC must be correctly configured to route Mobile Originated (MO) messages back to the OTA Manager V5.1 correct campaign module. This is discussed in “Step 5: Configuring Originating and Destination Addresses” that follows.

Step 5: Configuring Originating and Destination Addresses

If Processing PoRs

When running XCT campaigns that require a PoR to be returned by the subscriber's SIM card and unitary service requests in parallel on the same OTA Manager V5.1 platform, you must ensure that you assign different values of originating address (TP-OA) to each of the campaign modules.

To configure the originating address:

- For the Campaign Manager, set the ORIGINATING_ADDRESS product parameter.
- For XCT, specify the originating address in the TP_OA product parameter. Specify the TON/NPI in the TP_OA_TON_NPI product parameter.

Note: The TON/NPI for the destination address is specified in the TP_DA_TON_NPI product parameter.

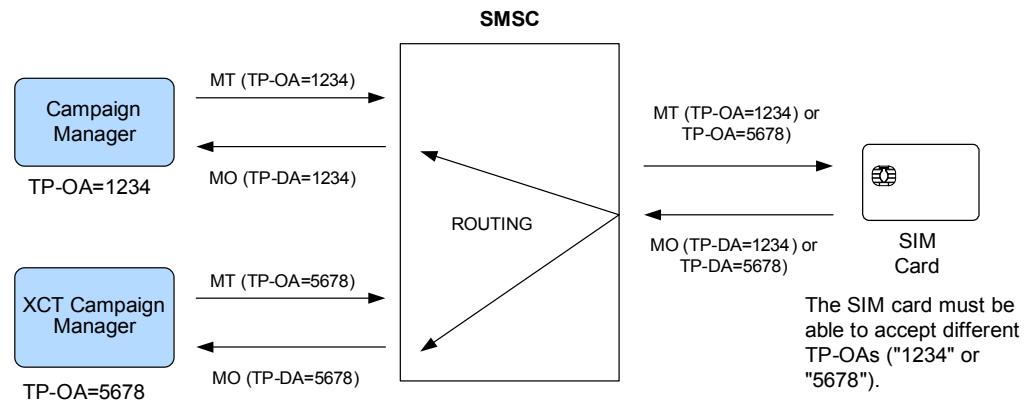
Refer to "Appendix B - Product Parameters" for details.

In addition, check that the subscriber's SIM card conforms to all relevant SIM card specifications:

- 1 The destination address (TP-DA) is not hard-coded in the SIM card.
- 2 If the SIM card maintains a list of valid TP-OAs, this list must be capable of holding at least two different addresses.
- 3 The SIM card must use the incoming TP-OA value as the TP-DA value in outgoing messages.

"Figure 54" illustrates this situation:

Figure 54 - TP-OA and TP-DA Addressing



If Not Processing PoRs

If no PoR processing is required, always configure XCT to use the same TP-OA value as that of the Card Manager (RCA).

To configure the originating address:

- For the Campaign Manager, set the ORIGINATING_ADDRESS product parameter.
- For XCT, the originating address is calculated by the Card Manager product performing the formatting (RCAForXCT), and XCT does not overwrite it.

If the originating address is not calculated within the service, then the default value specified in the PROTOCOLSMS/ORIGINATING_ADDRESS product parameter of RCAForXCT is used.

Depending on whether the originating address is international (starts with a "+") or national, the following RCAForXCT product parameter is used:

- PROTOCOLSMS/TONNPI_INT_ORIG
- PROTOCOLSMS/TONNPI_NAT_ORIG

Remarks:

- In this situation, the XCT product parameter TP_OA is not used.
- The TON/NPI for the destination address is configured in the XCT product parameter CHANNEL/TP_DA_TON_NPI.

Refer to "Appendix B - Product Parameters" for details.

Calculating the XCT Memory Footprint

The XCT memory footprint is the size of the memory image that is kept in the database for the XCT component. The XCT memory footprint depends on three parameters:

- A fixed minimum value. For XCT, this value is about 150 MB.
- The size of the invocation cache (PARAMETERS/INVOCATION_CACHE_SIZE product parameter). The invocation cache contains information kept in memory for each card. Its size has an important affect on performance.
- The size of the SMS cache (PARAMETERS/SMS_CACHE_SIZE product parameter). The SMS cache contains information kept in memory for each SMS message. This information is not used if the invocation only contains one SMS.

Calculating the Invocation Cache Memory Footprint

The size of the invocation cache (in bytes) can be calculated as follows:

$$\text{INVOCATION_CACHE_SIZE} * (260 + 130 * (\text{number_of_sms_per_card}))$$

Where:

- INVOCATION_CACHE_SIZE is the value of the XCT product parameter.
- 260 is a fixed value for the number of bytes used by an XCT invocation in the invocation cache.
- 130 is a fixed value for the size of a message in the invocation cache.
- *number_of_sms_per_card* is the real number of SMS messages, observed after the formatting phase.

Calculating the SMS Cache Memory Footprint

The size of the SMS cache (in bytes) can be calculated as follows:

$$\text{SMS_CACHE_SIZE} * 170$$

Where:

- SMS_CACHE_SIZE is the value of the XCT product parameter. This is the total number of SMS messages kept in memory.

Note that this parameter is independent from INVOCATION_CACHE_SIZE and the *number_of_sms_per_card*. The value can be less than or greater than $\text{INVOCATION_CACHE_SIZE} * \text{number_of_sms_per_card}$.

- 170 is a constant representing the size of an SMS message in the SMS cache.

Calculating the Global Memory Footprint

The global memory footprint (in bytes) for XCT is calculated as:

$$160\,000\,000 + \text{INVOCATION_CACHE_SIZE} * (260 + 130 * (\text{number_of_sms_per_card})) + \text{SMS_CACHE_SIZE} * 170$$

Where 160 000 000 is a constant (in bytes) representing the amount of memory occupied by XCT object instances in the Java Virtual Machine.

Note: The memory footprint must always be less than the value of the **XmX** value configured for the XCT product. **XmX** is a Java command line parameter that can be specified when launching XCT. The specified value is stored in the BS_COMPONENT table of the Framework database. Its default value is 384 (MB).

If this constraint is not respected, you might encounter “out of memory” errors when running campaigns because the caches are full. If so, reduce the values of the INVOCATION_CACHE_SIZE and SMS_CACHE_SIZE product parameters, or increase the value of the **XmX** command line parameter for XCT (provided there is sufficient memory available on the machine).

Monitoring XCT Campaigns

You can monitor XCT campaigns in four ways:

- Through the management interface of the Campaign Manager
- With SNMP traps and counters
- By using a traffic monitor tool such as Multi Router Traffic Grapher (MRTG) to provide statistics
- By analysing log files (required only for high-performance tuning)

Campaign Manager Management Interface

The Campaign Manager management interface enables you to view certain SMS and invocation statistics retrieved via SQL requests from the XCT database:

Three screens are available:

- Global statistics of XCT campaign
- Detailed statistics for the pre-processing, sending and post-processing stages of the campaign. Statistics can be global or sorted by target sets and details at SMS level are also available.
- Error statistics of XCT campaign. For each phase the number of invocations is given by error code and error detail. A sort by target set can also be specified.

Note: If an SMS of an invocation is FAILED, and if it is not the last SMS, the following SMS are not updated and are put in the state “ABORTED”. When an XCT campaign is “TERMINATED” or “COMPLETED” no SMS remain with “Not started” status.

SNMP Traps and Counters

Some SNMP traps and counters are available. They are described in “Appendix E - SNMP Counters, Alarms, and Traps” in this guide.

OTA Manager’s management interface also displays SNMP counters (**Product Info > Campaign Engines > XCT > Details**). Press F5 to refresh the window.

MRTG Statistics

You can use the Multi Router Traffic Grapher (MRTG) software package to generate XCT activity statistics from SNMP. MRTG is available under a GNU general public license. Version 2.11.1 of MRTG has been tested for compatibility and can be downloaded from:

<http://oss.oetiker.ch/mrtg/download.en.html>

Installing MRTG

- 1 Install MRTG by following the instructions in the MRTG documentation.
- 2 Copy the configuration file `mrtg.cfg` to the `mrtg-2.11.1\bin` directory.
- 3 In the configuration file `mrtg.cfg`, configure your output directory (`WorkDir`).
- 4 In the configuration file `mrtg.cfg`, enter your hostname and XCT product ID. Replace `<hostname>` with the name of your machine and `<xct_product_id>` with the identifier of your XCT product (found in the `N_COMP_ID` column of the `BS_COMPONENT` table in the Framework database).
- 5 In the configuration file `mrtg.cfg`, uncomment the corresponding lines if you have alerts (integration with BMD router) or SMS-MO (Proof Of Receipt).
- 6 Launch MRTG in daemon mode (explained in the MRTG documentation). The HTML pages are automatically generated every five minutes in your output directory.

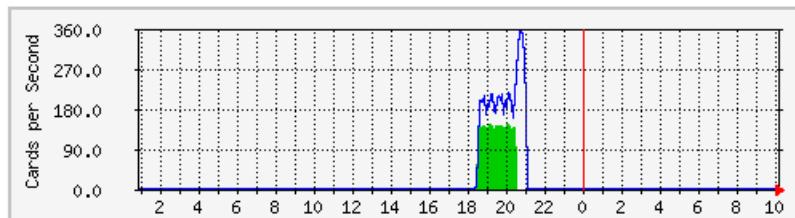
MRTG Examples

The following types of graphs can be generated using MRTG:

Pre- and post-processing throughput (cards/second):

Figure 55 - MRTG Showing Pre- and Post-Processing Throughput

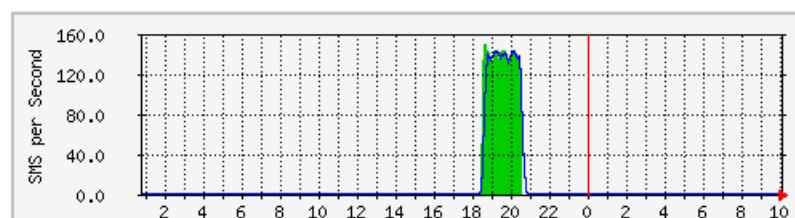
'Daily' Graph (5 Minute Average)



Sending throughput (SMS-MT and Delivery Status, in SMS/second):

Figure 56 - MRTG Showing Sending Throughput and Delivery Status

'Daily' Graph (5 Minute Average)



Other statistics can also be generated automatically:

- Alert throughput (alerts/sec) and SMS-MO throughput (MO/sec)
- Cache hit for invocation (%) and cache hit for SMS (%)
- Number of cards targeted in parallel.

Tuning Classic Campaigns

The Campaign Manager feature offers many configuration parameters. In order to use these parameters efficiently, it is important to adopt an initial strategy for the campaign to run, and adjust the settings according to the results obtained. Depending on the type of campaign you want to run, the tuning of these parameters is critical to a high success rate and a consistent SMSC bandwidth usage.

The spreadsheet `ClassicalCampaignsTuningWizard.xls` is provided to help support you in this task. The following description refers to this spreadsheet.

After having performed a number of small campaigns to validate the service's execution on SIM cards (no retries needed), bigger campaigns can be scheduled to analyze OTA Manager V5.1 and SMSC behavior when subject to heavy traffic loads. The idea is to find a proper threshold for every configuration in order to guarantee an optimized success rate.

The SMSC bandwidth is a very important element to take in account. This must be measured between the OTA Manager V5.1 platform and the SMSC.

Configuring classic campaign parameters involves:

1 Define campaign parameter values:

For parameters defined at integration time (see "Parameters Defined at Integration Time" on page 116), refer to:

- "Step 1: Determining the C_COMP_ARGS Value" on page 116.
- "Step 2: Determining the Memory Size (MEMORY_SIZE)" on page 116.
- "Step 3: Determining the Size of the SMSC Buffer (SWAP_SIZE)" on page 117.

For parameters defined in the CCI (see "Parameters Defined by the User" on page 118), refer to:

- "Step 4: Determining MIN_IN_MEMORY_TIME" on page 118.
- "Step 5: Choosing the SMS Transmission Mode" on page 118..
- "Step 6: Determining the Maximum Congestion (Max Congestion)" on page 118.
- "Step 7: Determining the Maximum Throughput (Max Throughput)" on page 120.
- "Step 8: Configuring the Validity Period" on page 120.
- "Step 9: Determining the Maximum Number of Retries" on page 121.
- "Step 10: Determining the Retry Delay" on page 121.

2 Update the parameter values:

- Campaign Manager product parameters (see "Updating Campaign Manager Module Product Parameters" on page 121).
- Campaign launch parameters "Updating a Campaign's Parameters" on page 122).

3 Launch the campaign and monitor its progress.

Based on monitoring observations and campaign results, you can then repeat these steps until the optimum values have been determined.

Parameters Defined at Integration Time

The following parameters are defined by Gemalto support personnel at installation time, and should not be modified by the user.

Step 1: Determining the C_COMP_ARGS Value

You must determine the **Xmx** (maximum memory) value to use for the Card Manager (RCA) used for the campaign.

The **Xmx** value is defined in the C_COMP_ARGS parameter in BS_COMPONENT table of the Framework database.

By default, this parameter is set to “**{ss2m,mx384m,noclassgc}**”, where **mx384m** specifies the maximum Java heap size: the Java Virtual Machine (JVM) can use up to a maximum of 384 MB of memory. This parameter must be set only on the Card Manager instance that is to run the campaigns (it is recommended to run all campaigns on a single Card Manager instance).

Refer to the Excel calculation file for help in calculating a suitable value.

The recommended values are:

- Small platform (less than 1 GB of RAM): **mx384m**
- Medium to large platform (between 1 GB and 4 GB RAM): **mx768m**
- Very large platform (more than 4 GB RAM): **mx1000m**

1 GB (“**mx1000m**”) is the maximum value to use on any platform. Above this limit, performance can decrease.

Step 2: Determining the Memory Size (MEMORY_SIZE)

The MEMORY_SIZE Card Manager product parameter specifies the amount of memory that a campaign occupies.

This parameter value must be set in accord with the **Xmx** value:

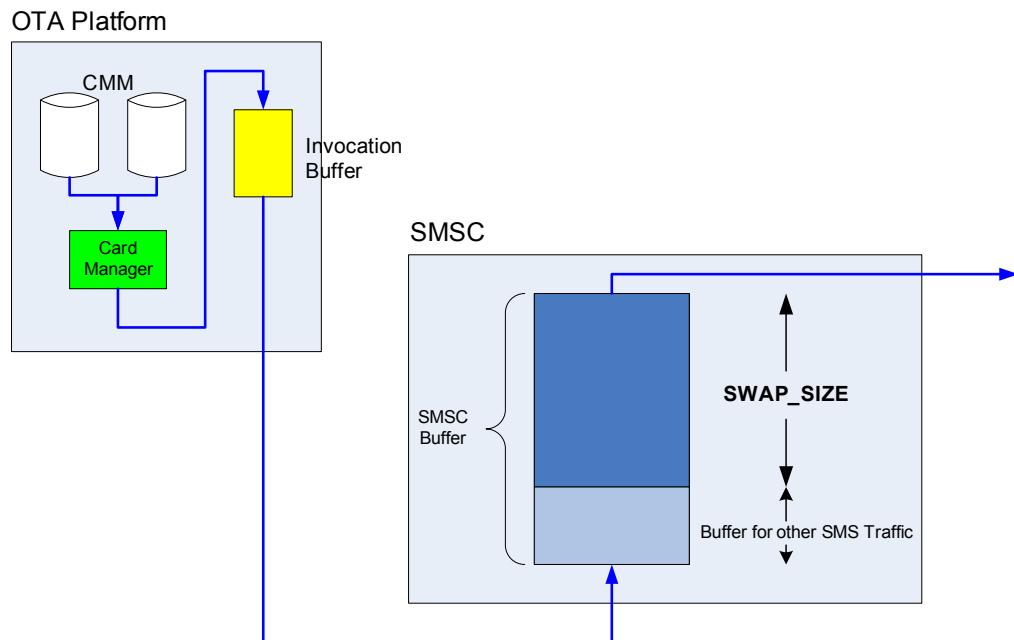
- If **Xmx** equals “384”, specify **1000**
- If **Xmx** equals “768”, specify **5000**
- If **Xmx** equals “1000”, specify **10000**

Warning: It is NOT recommended to modify this parameter’s value for individual campaigns. Only consider modifying the value if frequent “outOfMemory” errors occur, and only after consultation with Gemalto support personnel.

Step 3: Determining the Size of the SMSC Buffer (SWAP_SIZE)

The SWAP_SIZE Card Manager product parameter determines the size of the SMSC buffer that is reserved for the OTA platform's use.

Figure 57 - SWAP_SIZE: SMSC Buffer Reserved for OTA Traffic



The swap size determines the total number of invocations that can be stored in the SMSC during the validity period of services, the number of SMS messages originating from unitary service requests, the number of SMS messages automatically sent after card provisioning, and so on.

Warning: The SWAP_SIZE parameter is set by Gemalto during the integration phase.

As this parameter is linked to your use patterns (for example, the number of invocations in a campaign) and is associated with the Oracle database table size, you can fine tune this parameter but before doing so, you should obtain the agreement of Gemalto support personnel.

Risks:

- If the SWAP_SIZE parameter value is too high with respect to your OTA activities, the SMSC will have a storage problem, resulting in some SMS messages being deleted, which will in turn result in out of sequence messages.
- If the SWAP_SIZE parameter value is too low with respect to your OTA activities, you will unnecessarily restrict the SMS submission bandwidth.

Example

Total SMSC Buffer Size: 1,000,000 SMS

SWAP_SIZE: 600,000 SMS

SMS Buffer Remaining for Other SMS Traffic: 400,000 SMS

For details on how to change the parameter value, see “Updating Campaign Parameters” on page 121.

Parameters Defined by the User

You can update the following parameters for individual campaigns or campaign types.

Step 4: Determining MIN_IN_MEMORY_TIME

The MIN_IN_MEMORY_TIME Card Manager product parameter specifies the minimum amount of time that an SMS is retained in an area of RAM called the Swappable Storage Area (SSA) before being considered “out of coverage” and swapped to disk.

Setting the MIN_IN_MEMORY_TIME Parameter Value

The default value of the MIN_IN_MEMORY_TIME parameter is set by Gemalto during product integration.

In general it is calculated as:

Average time for Delivery Status reception for cards in coverage (generally between 10 and 40 seconds) + 20%

As this parameter depends on your SMSC and GSM network performance, you can fine tune this parameter's value:

- If the value of MIN_IN_MEMORY_TIME is too small for your GSM network's capabilities, it can significantly decrease campaign performance: invocations are first stored in RAM then on slower disk.
- If, however, the MIN_IN_MEMORY_TIME value is too high for your GSM network's capabilities, it can also significantly decrease campaign performance because invocations are stored in RAM then, when RAM is full, OTA Manager can no longer accept new invocations.

See the Excel spreadsheet “Ability to achieve wished SMSC bandwidth with current technical parameters” to check the capability to achieve the desired SMSC bandwidth with the current technical parameters. You can check this capability online on the Product Stats tab of the Campaign Monitoring window. The status should be green. If the status is yellow and the detail is “Output bandwidth can be affected by jobs flow control”, you should refine your tuning.

Gemalto support personnel can help you choose suitable values.

Step 5: Choosing the SMS Transmission Mode

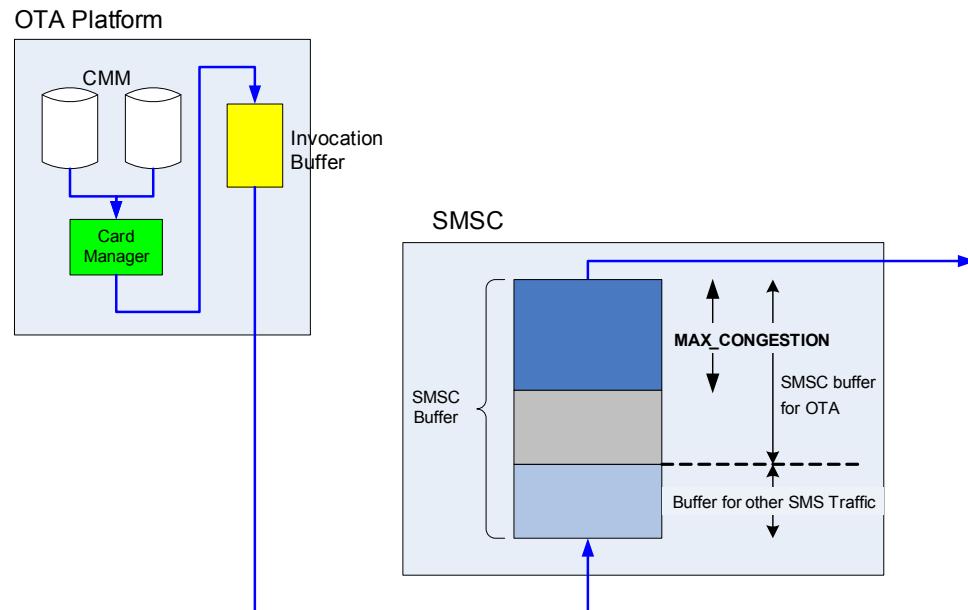
The Mode product parameter in the **PROTOCOLSMS** group of Card Manager product parameters allows you to choose the SMS mode to use: TRANS, MT, or SMSC.

You should always specify MT mode when submitting services and creating a campaign. Refer to the appropriate online help window.

Step 6: Determining the Maximum Congestion (Max Congestion)

Max Congestion is a Campaign Manager Module parameter value that determines the maximum number of invocations that can be sent by the OTA platform to the SMSC without receiving Delivery Status (DS) messages for these invocations.

This equates to the number of invocations stored in the SMSC buffer during the validity period of these invocations.

Figure 58 - Max Congestion

The default Max Congestion parameter value is set by Gemalto during the integration phase.

If the Max Congestion value is too high for your SMASH's capacity, the SMSC will have storage problems, and will delete some SMSs, which will then be out of order.

If the Max Congestion parameter is too low, you will unnecessarily restrict the SMS submission bandwidth.

Example

SMSC Buffer Size : 1,000,000 SMS

SMSC Buffer Size for OTA: 600,000 SMS

SMSC Buffer Size for Campaign (Max Congestion): 500,000 SMS

Spare SMS Buffer Size for Other SMS Traffic: 400,000 SMS

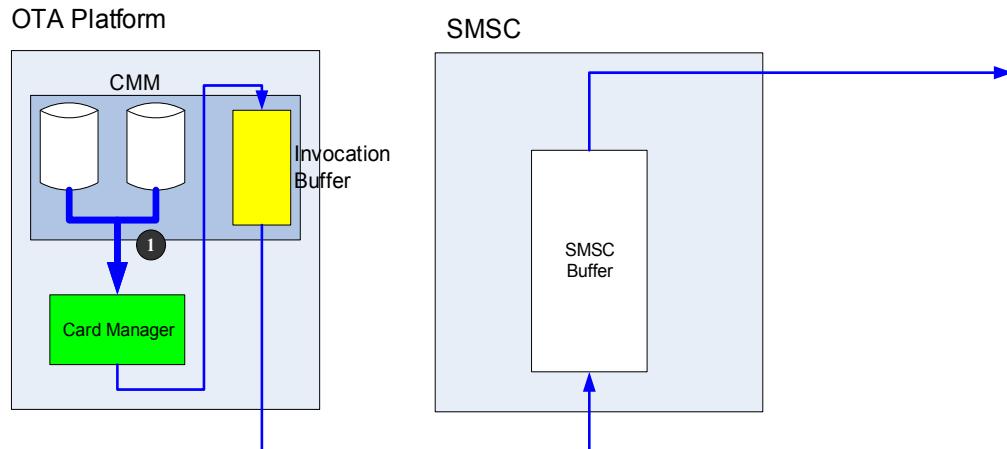
Gemalto support personnel can help you choose suitable values.

For details on how to change the parameter value, see “Updating Campaign Manager Module Product Parameters” on page 121.

Step 7: Determining the Maximum Throughput (Max Throughput)

Max Throughput is a Campaign Manager Module product parameter that determines the maximum number of requests sent by the Campaign Manager Module to the Card Manager every second (see ① in "Figure 59").

Figure 59 - Maximum Throughput



Use the Excel spreadsheet "Invocation Throughput" to set the maximum throughput.

Step 8: Configuring the Validity Period

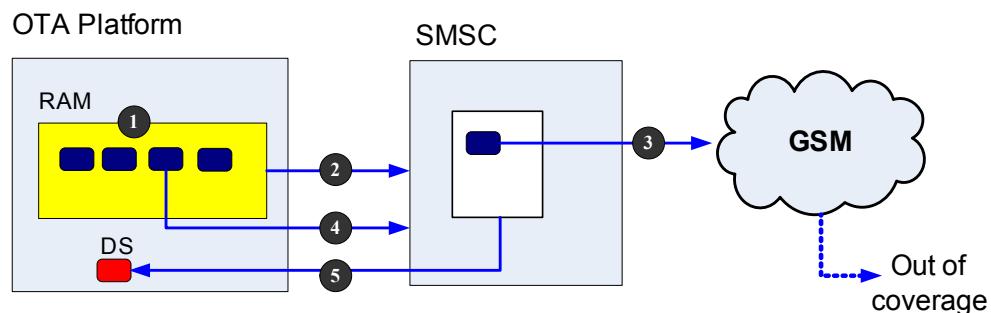
The validity period is a Campaign parameter that specifies the period of time within which all messages of an invocation must be successfully delivered for the service execution to be considered successful.

The validity period setting is crucial as it determines the success rate of a campaign.

The validity period value is passed to the SMSC, which sends the SMS messages during the entire validity period according to the SMSC retry scheme.

The validity period applies to each individual service: if a service requires the generation of more than one SMS message, each SMS of the service uses the same validity period value.

Figure 60 - The Validity Period



The validity period works as follows:

- ① An invocation is processed: SMS messages are generated by the OTA platform. The Validity Period of all of these SMS messages is the same and begins to expire as soon as the messages are created.
- ② The first SMS is sent to the SMSC.

- ③ The SMSC attempts to sent the SMS to the targeted card. If the targeted card is out of coverage, the SMS remains in the SMSC and a separate retry mechanism is initiated.
- ④ If the targeted card comes into GSM network coverage before the Validity Period expires, the next SMS of the service is sent to the SMSC. However, note that the Validity Period of all the SMS messages begins to expire as soon as the messages are generated.
- ⑤ Service execution is successful if all SMS messages of the service are successfully acknowledged and the DS is received by the card before the Validity Period expires.

The recommended Validity Period value is between 6 hours and 18 hours. The Excel spreadsheet “Max Validity Period” helps you check whether it is possible to set a particular value of validity period.

If the Validity Period is too short, a significant percentage of cards will not be updated.

If, however, the Validity Period is too long, you risk saturating the SMSC's buffer and reducing campaign performance.

For details on how to change the parameter value, see “Updating a Campaign’s Parameters” on page 122.

Step 9: Determining the Maximum Number of Retries

The Maximum Number of Retries parameter is a campaign parameter that specifies the number of attempts to make to execute a job that fails on a card after a certain amount of time. It allows you, for example, to target cards at a different time of the day. The retry mechanism is triggered by SIM Ack results, or it can be triggered by a specific error code of the SIM Nak defined in the EXPIRED_CODE_LIST field in the channel driver's `drivers.ini` file (see “Configuring Channel Drivers” on page 189).

Specify the number of times a service will be retried. 0 means it will be tried once only.

Step 10: Determining the Retry Delay

The Retry Delay is a campaign parameter that determines the amount of time (in seconds) before a retry is performed on a failed card. It is recommended to set the retry delay large enough to allow all the cards of the campaign be executed at least once.

Note: As retries are made on “expired” cards, they are likely to be made on “out of coverage” cards, which have previously failed on expiration (either they’re still in their box, or the terminal is off). If these cards have failed once, there is a high probability that future retries on the same cards will also fail, because they turn out to be out of coverage. For efficiency reasons, it is recommended to use zero retries, or to set a long retry delay otherwise, in order to let the good cards finish first, without the burden of looser cards

For details on how to change the parameter value, see “Updating a Campaign’s Parameters” on page 122.

Updating Campaign Parameters

Updating Campaign Manager Module Product Parameters

To update Campaign Manager Module product parameters:

- 1 Click **Platform** in the top menu.
- 2 Click **Product info.** on the left-hand menu.

- 3 Select the product (for example, “RCAForCMM”) in the **Registered Products** and click **Details**. The list of product parameters for the Campaign Manager Module displays in the Registered Product Parameters window:

Figure 61 - Campaign Manager Module Product Parameters

The screenshot shows a window titled "Registered Product Parameters". It contains four data entries:

- Max Throughput: 20
- Max Congestion: 3544
- Current Throughput: 0
- Current Congestion: 0

At the bottom right are two buttons: "Refresh" and "Apply".

- 4 Take note of the current values, then modify parameter values as required.
 5 Click **Apply**.

Updated values are taken into account immediately.

Updating a Campaign’s Parameters

Campaign processing parameters appear in the Create a New Campaign - Parameters Definition window (**Create Campaign > New**) when scheduling a new campaign:

Figure 62 - Campaign Parameters

The screenshot shows a window titled "Create a new campaign - Parameters definition". It has several sections:

- Scheduling Information**: Transaction identifier input field.
- Daily time information**: Three rows for "First frame", "Second frame", and "Third frame", each with "From" and "To" time inputs and clock icons.
- Processing parameters**:
 - Enable Monitoring of Campaign Invocations
 - Priority level: 50, between 1 and 100 (high priority)
 - Validity period / service: 600 seconds
 - Maximum number of retries: 3 times
 - Retry delay: 3600 seconds

Monitoring the Current Throughput and Congestion

Monitoring the Current Throughput

OTA Manager allows real-time monitoring of the Campaign Manager throughput. The **Current throughput** field on the Registered Product Parameters window (see “Figure 61” on page 122) shows the current value.

The **Max Throughput** product parameter is “hot”, that is, it can be updated dynamically during a campaign. Its value is therefore highly volatile, and should not be used to determine trends.

Monitoring Current Congestion

OTA Manager allows real-time monitoring of congestion. The **Current congestion** field on the Registered Product Parameters window (see “Figure 61” on page 122) shows the current value.

It is very useful to monitor the current congestion when a campaign appears to have frozen but is actually still executing. As explained previously, when congestion is at its maximum, the Campaign Manager waits for DS messages or expired events before sending new jobs to the Card Manager, thus preventing new jobs being created until these are received.

The **Max Congestion** product parameter is “hot”, that is, it can be updated dynamically during a campaign. Its value is therefore highly volatile, and should not be used to determine trends.

Monitoring the Success Rate and End of Campaigns

The platform offers a real-time monitoring with its current state and statistics:

- Total number of targeted cards.
- Total number of targeted cards either awaiting execution or currently executing.
- Total number of targeted cards executed with the following states: succeeded, failed and out of date.

At the end of a campaign’s execution (TERMINATED state), it is possible to generate an XML report:

- 1 Click **Monitoring > Details** in the management interface.
- 2 Select the desired campaign
- 3 Click on **Export** to generate the XML report.

The XML report describes the execution state of each card:

Table 12 - Campaign Report Status Codes

Status	Description
SUCCEEDED	The card has received all the SMS messages belonging to the service invocation.
FAILED	The card did not receive one or more of the SMS messages belonging to the service invocation and could therefore not execute the request. The error code returned in the SIM Ack or PoR indicates the origin of the error.
ABORTED	The campaign was aborted before all SMS messages could be sent to the targeted card.
OUT OF DATE	The platform did not receive the SIM Ack before the end of the validity period of the job. The end of the campaign is reached before the job’s creation.

The final results of a campaign can be determined by analysing the generated XML report, available for download in the management interface under **Monitoring > Report Files**.

For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<Campaign id="28" label="Campagne_Select_MF_GP91_3"
campaignScenarioId="4" campaignScenarioLabel="Service_Select_MF"
beginningDateMs="1080147122000" beginningDate="Wed Mar 24 18:52:02
```

```
GMT+02:00 2004" endingDateMs="1080149102000" endingDate="Wed Mar 24  
19:25:02 GMT+02:00 2004">  
<CardInformation MSISDN="97264444011" endingStatus="FAILED">  
<ServiceResult name="ExecScriptService" endingStatus="FAILED"  
errorCode="RCA-32" errorMessage="Card does not exist in Message Builder  
database or securityID is not coherent." numberOfWorkers="0"  
lastRetryDateMs="1080147122887" lastRetryDate="Wed Mar 24 18:52:02  
GMT+02:00 2004" completionDateMs="1080147124889" completionDate="Wed  
Mar  
24 18:52:04 GMT+02:00 2004"/>  
</CardInformation>  
<CardInformation MSISDN="97250348390" endingStatus="FAILED">  
<ServiceResult name="ExecScriptService" endingStatus="EXPIRED"  
errorCode="RCA-38" errorMessage="null" numberOfWorkers="0"  
lastRetryDateMs="1080147122887" lastRetryDate="Wed Mar 24 18:52:02  
GMT+02:00 2004" completionDateMs="1080148379887" completionDate="Wed  
Mar 24 19:12:59 GMT+02:00 2004"/>  
</CardInformation>  
<CardInformation MSISDN="97252499036" endingStatus="SUCCEEDED">  
<ServiceResult name="ExecScriptService" endingStatus="SUCCEEDED"  
numberOfWorkers="0" lastRetryDateMs="1080147122887" lastRetryDate="Wed  
Mar 24 18:52:02 GMT+02:00 2004" completionDateMs="1080147133900"  
completionDate="Wed Mar 24 18:52:13 GMT+02:00 2004"/>  
</CardInformation>  
</Campaign>
```

If the end status (`endingStatus` attribute) is SUCCEEDED, the operation was correctly executed on this card and does not need to be replayed in another campaign. For any other value of end status, the card is entered in a new campaign.

Reference Information

Batchloading XML Files

This chapter describes how to batchload the following objects into the Card Manager database:

- Card vendors (defines smart card vendors)
- Card definitions (defines the electronic characteristics of a card)
- Card profiles (defines a class of smart cards using other Card Manager entities)
- Service declarations and implementations
- Card instances (defines the association between individual smart cards and a card profile)
- Card security settings (defines the security settings for card instances)
- Group files
- Subscribers
- Classic campaign targets
- XCT campaign targets

Batchloading is used to create and modify large amounts of information, such as when populating the card database after product installation.

Order of Batchloading

The order in which objects are batchloaded is important. The order is:

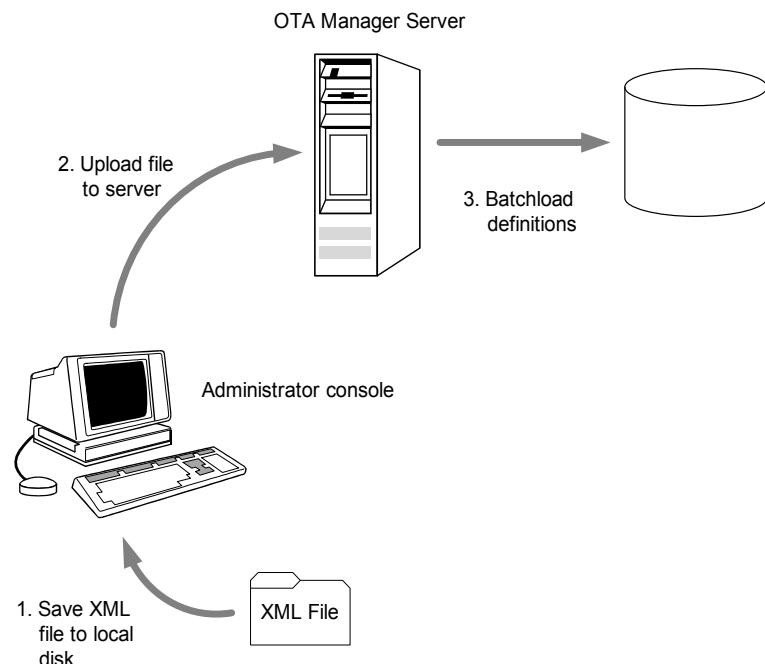
- 1 Card vendors
- 2 Card definitions
- 3 Card profiles
- 4 Service declarations
- 5 Service implementations
- 6 Card instances
- 7 Card security data

The Batchloading Process

To process card data in batch, you must create a file in an XML format that contains the new data.

To avoid overloading the network, OTA Manager V5.1 uses a local repository for the XML files, as shown in “Figure 63”:

Figure 63 - The Batchloading Process



Batchloading of card data takes place in several steps:

- 1 Store the XML file containing the new data to a local or networked hard disk, for example, on the administrator console.
- 2 On the appropriate page of the management interface, click the **Batchload** button and select the XML file. The file is uploaded to the OTA Manager V5.1 server machine.
- 3 OTA Manager V5.1 requests the database to update or create information in the database with the contents of the XML file.

To ensure that the data in the XML file is complete and consistent, it must comply with structure rules defined in an XML Document Type Definition (DTD). The DTD defines the structure of the documents, which items of information (*elements*) must or can be used, and any *attributes* associated with the elements.

OTA Manager V5.1 checks the XML file against the DTD rules at Step 3 above, that is, when parsing the XML file. If errors are found, the batchload fails and you must correct the errors that occur. A report is generated indicating the errors. You may find it useful to use a third-party tool to check the XML file against the DTD and correct any errors before batchloading the file.

Card Vendor Files

DTD Structure

```
<!-- Definition of Card Vendor Module -->
```

```
<!ELEMENT CardVendorModule (CardVendor)+>
<!ELEMENT CardVendor EMPTY>
<!ATTLIST CardVendor name CDATA #REQUIRED>
<!ATTLIST CardVendor description CDATA #IMPLIED>
```

The attributes of the `CardVendor` element used to represent a card vendor are as follows:

Table 13 - Card Vendor Parameters

Parameter	Description
name	Name of the card vendor (30 characters maximum).
description	Short description of the card vendor (80 characters maximum).

Example

The following example shows an XML file containing a single card vendor:

```
<?xml version="1.1" encoding="UTF-8"?>
<!DOCTYPE CardVendorModule SYSTEM "cardframework.dtd">
<CardVendorModule>
    <CardVendor
        name="Gemalto"
        description="Gemalto - La Ciotat, France" />
</CardVendorModule>
```

Management Interface Equivalent

Card Vendors can be provisioned in the management interface (**Card manager > Card > Vendor**). For example:

Figure 64 - Card Vendor XML and Management Interface

```
<CardVendorModule>
    <CardVendor
        name="Gemplus"
        description="Reference Data: vendor for Gemalto" />
</CardVendorModule>
```

Card definition description	
Card vendor name*	Gemplus
Description	Reference Data: vendor for Gem

All mandatory fields are specified by a *

Update Back

Card Definition Files

DTD Structure

```
<!-- Definition of Card Definition Module -->
<!ELEMENT CardDefinitionModule (CardDefinition)+>
<!ELEMENT CardDefinition EMPTY>
<!ATTLIST CardDefinition internalName CDATA #REQUIRED>
<!ATTLIST CardDefinition commercialName CDATA #IMPLIED>
<!ATTLIST CardDefinition osName CDATA #IMPLIED>
```

```
<!ATTLIST CardDefinition osVersion CDATA #IMPLIED>
<!ATTLIST CardDefinition chipManufacturer CDATA #IMPLIED>
<!ATTLIST CardDefinition chipModel CDATA #IMPLIED>
<!ATTLIST CardDefinition eepromSize CDATA #IMPLIED>
<!ATTLIST CardDefinition ramSize CDATA #IMPLIED>
<!ATTLIST CardDefinition romSize CDATA #IMPLIED>
<!ATTLIST CardDefinition freeSize CDATA #IMPLIED>
```

The DTD structure used to represent Card Definition data consists of the following parameters:

Table 14 - Card Definition Parameters

Parameter	Description
internalName	Internal name defined in the <i>SIM Card Product Management Group</i> document (30 characters maximum).
commercialName	Commercial name defined in the <i>SIM Card Product Management Group</i> document (30 characters maximum).
osName	Name of embedded operating system (30 characters maximum).
osVersion	Version of embedded operating system (10 characters maximum).
chipManufacturer	Name of the chip maker (30 characters maximum).
chipModel	Model reference of the chip (30 characters maximum).
eepromSize	Electric Erasable Programmable ROM size (in bytes).
ramSize	RAM size (in bytes).
romSize	ROM size (in bytes).
freeSize	Amount of free memory in smart card after personalization (in bytes).

Example

The following example shows an XML file representing a Card Definition entry:

```
<!DOCTYPE CardDefinitionModule SYSTEM "cardframework.dtd">
<CardDefinitionModule>
    <CardDefinition
        internalName="GEMX16"
        commercialName="GemXplore98"
        osName="GemXplore 07"
        osVersion="07.00"
        chipManufacturer="Siemens"
        chipModel="SLE66C"
        eepromSize="3200"
        ramSize="32000"
        romSize="32000"
        freeSize="64000" />
</CardDefinitionModule>
```

Management Interface Equivalent

Card Definitions can be provisioned in the management interface (**Card manager > Card > Definition**). For example:

Figure 65 - Card Definition XML and Management Interface

```
<CardDefinitionModule>
  <CardDefinition
    internalName="GX3G_v2.2_128K"
    commercialName="GemXplore 3G v2.2 128K"
    osName="GemXplore3GV2.2"
    osVersion="MSA04"
    chipManufacturer="Infineon"
    chipModel="Confidential"
    eepromSize="65536"
    ramSize="-1"
    romSize="-1"
    freeSize="-1" />
</CardDefinitionModule>
```

Card definition description	
Card definition name*	GX3G_v2.2_128K
Commercial name	GemXplore 3G v2.2 128K
OS name*	GemXplore3GV2.2
OS version	MSA04
Chip manufacturer	Infineon
Chip model	Confidential
EEPROM size	65536
RAM size	-1
ROM size	-1
Free size	-1

All mandatory fields are specified by a *

[Back](#)

Card Profile Files

The Card Profile references data from other Card Manager objects, including:

- Card vendors
- Card definitions
- Service implementations

The attributes of the `CardProfile` element used to describe a card profile are as follows:

Table 15 - Card Profile Parameters

Parameter	Description
name	Name of card profile (30 characters maximum).
cardVendor	Card manufacturer (30 characters maximum).
cardDefinition	Internal card definition name.

Table 15 - Card Profile Parameters

Parameter	Description
serviceImpl	Associates a card profile with a list of service implementations. Each service implementation is separated by a semi-colon (";").
state	State of the profile. Valid values: "0" (New), "1" (Inactive), or "2" (Active).
phase	Phase of the card.
capabilities	List of supported protocols: ESMS V1, ESMS V2, 03.48, CAT-TP

DTD Structure

```

<!-- Definition of Card Profile Module -->
<!ELEMENT CardProfileModule (CardProfile)+>
<!ELEMENT CardProfile (CardVendor?, CardDefinition?, ServiceImpl?,
FormattingProperties?, TransportProperties?, EntityProperties?,
CardStructure?)>
<!ATTLIST CardProfile name CDATA #REQUIRED>
<!ATTLIST CardProfile cardVendor CDATA #IMPLIED>
<!ATTLIST CardProfile cardDefinition CDATA #IMPLIED>
<!ATTLIST CardProfile serviceImpl CDATA #IMPLIED>
<!ATTLIST CardProfile state CDATA #IMPLIED>
<!ATTLIST CardProfile phase CDATA #IMPLIED>
<!ATTLIST CardProfile capabilities CDATA #IMPLIED>
<!-- List of capabilities (ESMSV1;ESMSV2;0348;BIP;CAT-TP) -->

<!-- Definition of Card Vendor Module -->
<!ELEMENT CardVendorModule (CardVendor)+>
<!ELEMENT CardVendor EMPTY>
<!ATTLIST CardVendor name CDATA #REQUIRED>
<!ATTLIST CardVendor description CDATA #IMPLIED>

<!-- Definition of Card Definition Module -->
<!ELEMENT CardDefinitionModule (CardDefinition)+>
<!ELEMENT CardDefinition EMPTY>
<!ATTLIST CardDefinition internalName CDATA #REQUIRED>
<!ATTLIST CardDefinition commercialName CDATA #IMPLIED>
<!ATTLIST CardDefinition osName CDATA #REQUIRED>
<!ATTLIST CardDefinition osVersion CDATA #IMPLIED>
<!ATTLIST CardDefinition chipManufacturer CDATA #IMPLIED>
<!ATTLIST CardDefinition chipModel CDATA #IMPLIED>
<!ATTLIST CardDefinition eepromSize CDATA #IMPLIED>
<!ATTLIST CardDefinition ramSize CDATA #IMPLIED>
<!ATTLIST CardDefinition romSize CDATA #IMPLIED>
<!ATTLIST CardDefinition freeSize CDATA #IMPLIED>

<!-- General entities used for file type definition -->
<!ENTITY NO_SECURITY_LEVEL "-1">

<!-- Definition of Card Service Implementation Module -->
<!ELEMENT ServiceImplModule (ServiceImpl)+>
<!ELEMENT ServiceImpl (SMS03_40?)>
```

```
<!ATTLIST ServiceImpl name CDATA #REQUIRED>
<!ATTLIST ServiceImpl serviceName CDATA #REQUIRED>
<!ATTLIST ServiceImpl javaClassName CDATA #IMPLIED>
<!ATTLIST ServiceImpl scriptName CDATA #IMPLIED>
<!ATTLIST ServiceImpl state CDATA #IMPLIED>
<!ATTLIST ServiceImpl cardContentUpdateActivated (true|false)
#REQUIRED>
<!ATTLIST ServiceImpl cardContentCheckActivated (true|false) #IMPLIED>
<!ATTLIST ServiceImpl applicationIdentifier CDATA #IMPLIED>
<!ATTLIST ServiceImpl securityLevel CDATA #IMPLIED>

<!ELEMENT SMS03_40 EMPTY>
<!ATTLIST SMS03_40 pid CDATA #REQUIRED>
<!ATTLIST SMS03_40 dcs CDATA #REQUIRED>

<!-- Definition of Card Service Declaration Module -->
<!ELEMENT ServiceDeclModule (ServiceDecl)+>
<!ELEMENT ServiceDecl EMPTY>
<!ATTLIST ServiceDecl name CDATA #REQUIRED>
<!ATTLIST ServiceDecl handlerName CDATA #REQUIRED>
<!ATTLIST ServiceDecl state CDATA #IMPLIED>
<!ATTLIST ServiceDecl isCardIndependent (true|false) #REQUIRED>

<!-- General entities used for file type definition -->
<!ENTITY MF_FILE "0">
<!ENTITY DF_FILE "1">
<!ENTITY EF_TRANSPARENT_FILE "2">
<!ENTITY EF_LINEAR_FILE "3">
<!ENTITY EF_CYCLIC_FILE "4">

<!-- Definition of Card Structure Module -->
<!ELEMENT CardStructureModule (CardStructure)+>
<!ELEMENT CardStructure (CardFile, JavaPackage*, NativeApplication*)>
<!ATTLIST CardStructure name CDATA #REQUIRED>

<!-- Definition of the Card File -->
<!ELEMENT CardFile (CardFile*)>
<!ATTLIST CardFile identifier CDATA #REQUIRED>
<!ATTLIST CardFile name CDATA #IMPLIED>
<!ATTLIST CardFile type CDATA #IMPLIED>
<!ATTLIST CardFile fileSize CDATA #IMPLIED>
<!ATTLIST CardFile invalidate CDATA #IMPLIED>
<!ATTLIST CardFile recordNumber CDATA #IMPLIED>
<!ATTLIST CardFile recordSize CDATA #IMPLIED>
<!ATTLIST CardFile defaultData CDATA #IMPLIED>
<!ATTLIST CardFile appliNumber CDATA #IMPLIED>
<!ATTLIST CardFile securityDomain CDATA #IMPLIED>

<!-- Definition of the JavaPackage -->
<!ELEMENT JavaPackage (JavaApplet*)>
```

```

<!ATTLIST JavaPackage aid CDATA #REQUIRED>
<!ATTLIST JavaPackage majorVersion CDATA #REQUIRED>
<!ATTLIST JavaPackage minorVersion CDATA #REQUIRED>
<!ATTLIST JavaPackage estimatedSize CDATA #REQUIRED>
<!ATTLIST JavaPackage securityDomain CDATA #IMPLIED>

<!-- Definition of the JavaApplet -->
<!ELEMENT JavaApplet (JavaAppletInstance*)>
<!ATTLIST JavaApplet aid CDATA #REQUIRED>

<!-- Definition of the JavaAppletInstance -->
<!ELEMENT JavaAppletInstance EMPTY>
<!ATTLIST JavaAppletInstance aid CDATA #REQUIRED>
<!ATTLIST JavaAppletInstance status (INSTALLED|LOCKED_FROM_INSTALLED|
SELECTABLE|LOCKED_FROM_SELECTABLE|PERSONALIZED|
LOCKED_FROM_PERSONALIZED) #REQUIRED>
<!ATTLIST JavaAppletInstance estimatedSize CDATA #REQUIRED>
<!ATTLIST JavaAppletInstance configuration CDATA #IMPLIED>
<!ATTLIST JavaAppletInstance securityDomain CDATA #IMPLIED>
<!ATTLIST JavaAppletInstance tar CDATA #IMPLIED>
<!ATTLIST JavaAppletInstance privileges CDATA #IMPLIED>
<!ATTLIST JavaAppletInstance isIssuerSD (true|false) #IMPLIED>
<!ATTLIST JavaAppletInstance ownerIdentificationNumber CDATA #IMPLIED>

<!-- Definition of the NativeApplication -->
<!ELEMENT NativeApplication (NativeService*)>
<!ATTLIST NativeApplication aid CDATA #REQUIRED>
<!ATTLIST NativeApplication majorVersion CDATA #REQUIRED>
<!ATTLIST NativeApplication minorVersion CDATA #REQUIRED>
<!ATTLIST NativeApplication status (INSTALLED|LOCKED_FROM_INSTALLED)#
REQUIRED>
<!ATTLIST NativeApplication estimatedSize CDATA #REQUIRED>
<!ATTLIST NativeApplication configuration CDATA #IMPLIED>

<!-- Definition of the NativeService -->
<!ELEMENT NativeService EMPTY>
<!ATTLIST NativeService aid CDATA #REQUIRED>
<!ATTLIST NativeService version CDATA #REQUIRED>
<!ATTLIST NativeService status (INSTALLED|LOCKED_FROM_INSTALLED)#
REQUIRED>
<!ATTLIST NativeService estimatedSize CDATA #REQUIRED>
<!ATTLIST NativeService configuration CDATA #IMPLIED>

<!-- Definition of the list of entity properties -->
<!ELEMENT EntityProperties (DirectoryFileProperties*,|
JavaPackageProperties*, JavaAppletProperties*,|
JavaAppletInstanceProperties*, NativeApplicationProperties*,|
NativeServiceProperties*)>

<!-- Definition of the EntityProperties -->
<!ELEMENT DirectoryFileProperties (SecurisationSet?)>
<!ATTLIST DirectoryFileProperties identifier CDATA #REQUIRED>

```

```

<!ELEMENT NativeApplicationProperties EMPTY>
<!ATTLIST NativeApplicationProperties aid CDATA #REQUIRED>
<!ATTLIST NativeApplicationProperties majorVersion CDATA #REQUIRED>
<!ATTLIST NativeApplicationProperties minorVersion CDATA #REQUIRED>
<!ATTLIST NativeApplicationProperties df CDATA #REQUIRED>
<!ATTLIST NativeApplicationProperties label CDATA #IMPLIED>

<!ELEMENT NativeServiceProperties EMPTY>
<!ATTLIST NativeServiceProperties appliAid CDATA #REQUIRED>
<!ATTLIST NativeServiceProperties appliMajorVersion CDATA #REQUIRED>
<!ATTLIST NativeServiceProperties appliMinorVersion CDATA #REQUIRED>
<!ATTLIST NativeServiceProperties serviceAid CDATA #REQUIRED>
<!ATTLIST NativeServiceProperties version CDATA #REQUIRED>
<!ATTLIST NativeServiceProperties label CDATA #IMPLIED>

<!ELEMENT JavaPackageProperties (SecurisationSet?)>
<!ATTLIST JavaPackageProperties packageAid CDATA #REQUIRED>
<!ATTLIST JavaPackageProperties packageMajorVersion CDATA #REQUIRED>
<!ATTLIST JavaPackageProperties packageMinorVersion CDATA #REQUIRED>
<!ATTLIST JavaPackageProperties label CDATA #IMPLIED>

<!ELEMENT JavaAppletProperties (SecurisationSet?)>
<!ATTLIST JavaAppletProperties appletAid CDATA #REQUIRED>
<!ATTLIST JavaAppletProperties packageAid CDATA #REQUIRED>
<!ATTLIST JavaAppletProperties packageMajorVersion CDATA #REQUIRED>
<!ATTLIST JavaAppletProperties packageMinorVersion CDATA #REQUIRED>
<!ATTLIST JavaAppletProperties label CDATA #IMPLIED>
<!ATTLIST JavaAppletProperties concatenationAvailable (true|false)
#REQUIRED>
<!ATTLIST JavaAppletProperties isSelectable (true|false) #REQUIRED>

<!ELEMENT JavaAppletInstanceProperties (SecurisationSet?)>
<!ATTLIST JavaAppletInstanceProperties instanceAid CDATA #REQUIRED>
<!ATTLIST JavaAppletInstanceProperties appletAid CDATA #REQUIRED>
<!ATTLIST JavaAppletInstanceProperties packageAid CDATA #REQUIRED>
<!ATTLIST JavaAppletInstanceProperties packageMajorVersion CDATA
#REQUIRED>
<!ATTLIST JavaAppletInstanceProperties packageMinorVersion CDATA
#REQUIRED>
<!ATTLIST JavaAppletInstanceProperties label CDATA #IMPLIED>
<!ATTLIST JavaAppletInstanceProperties msl CDATA #IMPLIED>
<!ATTLIST JavaAppletInstanceProperties cattpPort CDATA #IMPLIED>

<!-- Definition of the list of formatting properties -->
<!ELEMENT FormattingProperties (Formatting0348Properties?,,
FormattingGPPProperties?)>

<!-- Definition of a 0348 formatting properties -->
<!ELEMENT Formatting0348Properties EMPTY>

```

```

<!ATTLIST Formatting0348Properties isCplChlRequiredInCertificate (true|
false) #REQUIRED>
<!ATTLIST Formatting0348Properties rpFieldRequiredInCertificate
(RP_CERTIF_ALL | RP_CERTIF_WITHOUT_RPI | RP_CERTIF_WITHOUT_RPI_RPL_RHL)
#IMPLIED>
<!ATTLIST Formatting0348Properties isRplRhlRequiredInCertificate CDATA
#IMPLIED> <!-- DEPRECATED use rpFieldRequiredInCertificate -->
<!ATTLIST Formatting0348Properties isRpIRequiredInCertificate CDATA
#IMPLIED> <!-- DEPRECATED use rpFieldRequiredInCertificate -->

<!-- Definition of a GP formatting properties -->
<!ELEMENT FormattingGPProperties EMPTY>
<!ATTLIST FormattingGPProperties multipleLoadCommandPerMessage CDATA
#REQUIRED>
<!ATTLIST FormattingGPProperties maxLoadCommandLength CDATA #REQUIRED>
<!ATTLIST FormattingGPProperties tokenKeyVersion CDATA #IMPLIED >
<!ATTLIST FormattingGPProperties tokenKeyIndex CDATA #IMPLIED >
<!ATTLIST FormattingGPProperties receiptKeyVersion CDATA #IMPLIED >
<!ATTLIST FormattingGPProperties receiptKeyIndex CDATA #IMPLIED >

<!-- Definition of the list of transport properties -->
<!ELEMENT TransportProperties (TransportProperties0340?)>

<!-- Definition of the TransportProperties0340 -->
<!ELEMENT TransportProperties0340 EMPTY>
<!ATTLIST TransportProperties0340 tpudBufferSize CDATA #REQUIRED>
<!ATTLIST TransportProperties0340 concatBufferSpace CDATA #REQUIRED>

<!-- Definition of the Card Content and Smart Cards Module -->
<!ELEMENT CardContentModule (SmartCard*, SimCard*)>
<!ATTLIST CardContentModule profileName CDATA #IMPLIED>
<!ELEMENT SmartCard (CardFile*)>
<!ATTLIST SmartCard serialNumber CDATA #REQUIRED>
<!ATTLIST SmartCard profileName CDATA #IMPLIED>
<!ATTLIST SmartCard state CDATA #IMPLIED>

<!ELEMENT SimCard (CardFile*, JavaPackage*, JavaAppletInstance*, NativeApplication*, NativeService*)>
<!ATTLIST SimCard serialNumber CDATA #REQUIRED>
<!ATTLIST SimCard profileName CDATA #IMPLIED>
<!ATTLIST SimCard state CDATA #IMPLIED>
<!ATTLIST SimCard imsi CDATA #IMPLIED>
<!ATTLIST SimCard msisdn CDATA #IMPLIED>
<!ATTLIST SimCard linkedCard CDATA #IMPLIED>
<!ATTLIST SimCard freeEEPROMSizeCDATA #IMPLIED>
<!ATTLIST SimCard freeVolatileSizeCDATA #IMPLIED>
<!ATTLIST SimCard freeNonVolatileSizeCDATA #IMPLIED>
<!ATTLIST SimCard checkSecurity (true|false) #IMPLIED>

<!-- Definition of securisation -->
<!ELEMENT SecurisationSet (Securisation0348*|SecurisationNative*| SecurisationEsmsV1*|SecurisationEsmsV2*)>

```

```

<!ELEMENT Securisation0348 EMPTY>
<!ATTLIST Securisation0348 index CDATA #REQUIRED>
<!ATTLIST Securisation0348 spi CDATA #REQUIRED>
<!ATTLIST Securisation0348 kic CDATA #REQUIRED>
<!ATTLIST Securisation0348 kid CDATA #REQUIRED>

<!ELEMENT SecurisationNative EMPTY>
<!ATTLIST SecurisationNative index CDATA #REQUIRED>
<!ATTLIST SecurisationNative certifKeyNumber CDATA #REQUIRED>
<!ATTLIST SecurisationNative cipherKeyNumber CDATA #REQUIRED>

<!ELEMENT SecurisationEsmsV1 EMPTY>
<!ATTLIST SecurisationEsmsV1 index CDATA #REQUIRED>
<!ATTLIST SecurisationEsmsV1 systemFileId CDATA #REQUIRED>
<!ATTLIST SecurisationEsmsV1 certifKeyNumber CDATA #REQUIRED>
<!ATTLIST SecurisationEsmsV1 cipherKeyNumber CDATA #REQUIRED>

<!ELEMENT SecurisationEsmsV2 EMPTY>
<!ATTLIST SecurisationEsmsV2 index CDATA #REQUIRED>
<!ATTLIST SecurisationEsmsV2 app CDATA #REQUIRED>
<!ATTLIST SecurisationEsmsV2 sec CDATA #REQUIRED>
<!ATTLIST SecurisationEsmsV2 certifKeyNumber CDATA #REQUIRED>
<!ATTLIST SecurisationEsmsV2 cipherKeyNumber CDATA #REQUIRED>

<!ELEMENT CardGroupList (CardGroup)*>
<!ELEMENT CardGroup (Card)*>
<!ATTLIST CardGroup id CDATA #REQUIRED>
<!ATTLIST CardGroup description CDATA #REQUIRED>

<!ELEMENT Card EMPTY>
<!ATTLIST Card iccid CDATA #IMPLIED>
<!ATTLIST Card msisdn CDATA #IMPLIED>
<!ATTLIST Card imsi CDATA #IMPLIED>

```

Example

```

<!DOCTYPE CardProfileModule SYSTEM "cardframework.dtd">

<CardProfileModule>
    <CardProfile name="mNFC_v1.0"
        state="1"
        phase="2+"
        cardVendor="Gemplus"
        cardDefinition="mNFC_v1.0"
        serviceImpl=""
        capabilities="0348;CATPP">
        <FormattingProperties>
            <Formatting0348Properties isCplChlRequiredInCertificate="true"
                rpFieldRequiredInCertificate="RP_CERTIF_ALL"/>
            <FormattingGPPProperties multipleLoadCommandPerMessage="true"
                maxLoadCommandLength="255"
                tokenKeyVersion="70"

```

```
        tokenKeyIndex="1"
        receiptKeyVersion="71"
        receiptKeyIndex="1"/>
    </FormattingProperties>
    <TransportProperties>
        <TransportProperties0340 tpudBufferSize="140"
            concatBufferSpace="255"/>
    </TransportProperties>
    <EntityProperties>
        <JavaPackageProperties packageAid="A000000018200100"
            packageMajorVersion="1"
            packageMinorVersion="0"
            label="SD package"/>
        <JavaPackageProperties packageAid="A00000001820010106"
            packageMajorVersion="1"
            packageMinorVersion="0"
            label="System package"/>
        <JavaPackageProperties packageAid="A00000001820010302"
            packageMajorVersion="1"
            packageMinorVersion="0"
            label="RFM package"/>
        <JavaAppletProperties appletAid="A00000001820010002"
            packageAid="A000000018200100"
            packageMajorVersion="1"
            packageMinorVersion="0"
            label="SD applet"
            concatenationAvailable="true"
            isSelectable="true"/>
        <JavaAppletProperties appletAid="A0000000182001010300"
            packageAid="A00000001820010106"
            packageMajorVersion="1"
            packageMinorVersion="0"
            label="UICC applet"
            concatenationAvailable="true"
            isSelectable="true"/>
        <JavaAppletProperties appletAid="A0000000182001010301"
            packageAid="A00000001820010106"
            packageMajorVersion="1"
            packageMinorVersion="0"
            label="SIM applet"
            concatenationAvailable="true"
            isSelectable="true"/>
        <JavaAppletProperties appletAid="A0000000182001010302"
            packageAid="A00000001820010106"
            packageMajorVersion="1"
            packageMinorVersion="0"
            label="USIM applet"
            concatenationAvailable="true"
            isSelectable="true"/>
        <JavaAppletProperties appletAid="A0000000182001030201"
            packageAid="A00000001820010302"
            packageMajorVersion="1"
            packageMinorVersion="0"
            label="RFM applet"
            concatenationAvailable="true"
            isSelectable="true"/>
```

```
<JavaAppletInstanceProperties
    instanceAid="A000000018434DFF33FFFF89000000"
    appletAid="A00000001820010002"
    packageAid="A000000018200100"
    packageMajorVersion="1"
    packageMinorVersion="0"
    label="Card manager"
    cattpPort="100">
<SecurisationSet>
    <Securisation0348 index="6" spi="1221" kic="00" kid="10"/>
    <Securisation0348 index="5" spi="1200" kic="00" kid="10"/>
    <Securisation0348 index="4" spi="0621" kic="10" kid="10"/>
    <Securisation0348 index="3" spi="0221" kic="00" kid="10"/>
    <Securisation0348 index="2" spi="0021" kic="00" kid="00"/>
    <Securisation0348 index="1" spi="0000" kic="00" kid="00"/>
    <Securisation0348 index="0" spi="0000" kic="00" kid="00"/>
</SecurisationSet>
</JavaAppletInstanceProperties>
<JavaAppletInstanceProperties
    instanceAid="A000000018434DFF33FFFF89C00000"
    appletAid="A00000001820010002"
    packageAid="A000000018200100"
    packageMajorVersion="1"
    packageMinorVersion="0"
    label="SSD1"
    cattpPort="101">
<SecurisationSet>
    <Securisation0348 index="6" spi="1221" kic="00" kid="10"/>
    <Securisation0348 index="5" spi="1200" kic="00" kid="10"/>
    <Securisation0348 index="4" spi="0621" kic="10" kid="10"/>
    <Securisation0348 index="3" spi="0221" kic="00" kid="10"/>
    <Securisation0348 index="2" spi="0021" kic="00" kid="00"/>
    <Securisation0348 index="1" spi="0000" kic="00" kid="00"/>
    <Securisation0348 index="0" spi="0000" kic="00" kid="00"/>
</SecurisationSet>
</JavaAppletInstanceProperties>
<JavaAppletInstanceProperties
    instanceAid="A000000018434DFF33FFFF89C00001"
    appletAid="A00000001820010002"
    packageAid="A000000018200100"
    packageMajorVersion="1"
    packageMinorVersion="0"
    label="SSD2"
    cattpPort="102">
<SecurisationSet>
    <Securisation0348 index="6" spi="1221" kic="00" kid="10"/>
    <Securisation0348 index="5" spi="1200" kic="00" kid="10"/>
    <Securisation0348 index="4" spi="0621" kic="10" kid="10"/>
    <Securisation0348 index="3" spi="0221" kic="00" kid="10"/>
    <Securisation0348 index="2" spi="0021" kic="00" kid="00"/>
    <Securisation0348 index="1" spi="0000" kic="00" kid="00"/>
    <Securisation0348 index="0" spi="0000" kic="00" kid="00"/>
</SecurisationSet>
</JavaAppletInstanceProperties>
<JavaAppletInstanceProperties
    instanceAid="A000000018434DFF33FFFF89C00002"
```

```
        appletAid="A00000001820010002"
        packageAid="A000000018200100"
        packageMajorVersion="1"
        packageMinorVersion="0"
        label="SSD3"
        cattpPort="103">
        <SecurisationSet>
            <Securisation0348 index="6" spi="1221" kic="00" kid="10"/>
            <Securisation0348 index="5" spi="1200" kic="00" kid="10"/>
            <Securisation0348 index="4" spi="0621" kic="10" kid="10"/>
            <Securisation0348 index="3" spi="0221" kic="00" kid="10"/>
            <Securisation0348 index="2" spi="0021" kic="00" kid="00"/>
            <Securisation0348 index="1" spi="0000" kic="00" kid="00"/>
            <Securisation0348 index="0" spi="0000" kic="00" kid="00"/>
        </SecurisationSet>
    </JavaAppletInstanceProperties>
    <JavaAppletInstanceProperties
        instanceAid="A000000018434DFF33FFFF89C00003"
        appletAid="A00000001820010002"
        packageAid="A000000018200100"
        packageMajorVersion="1"
        packageMinorVersion="0"
        label="SSD4"
        cattpPort="104">
        <SecurisationSet>
            <Securisation0348 index="6" spi="1221" kic="00" kid="10"/>
            <Securisation0348 index="5" spi="1200" kic="00" kid="10"/>
            <Securisation0348 index="4" spi="0621" kic="10" kid="10"/>
            <Securisation0348 index="3" spi="0221" kic="00" kid="10"/>
            <Securisation0348 index="2" spi="0021" kic="00" kid="00"/>
            <Securisation0348 index="1" spi="0000" kic="00" kid="00"/>
            <Securisation0348 index="0" spi="0000" kic="00" kid="00"/>
        </SecurisationSet>
    </JavaAppletInstanceProperties>
    <JavaAppletInstanceProperties
        instanceAid="A0000000871001FF33FFFF8901010100"
        appletAid="A0000000182001010300"
        packageAid="A00000001820010106"
        packageMajorVersion="1"
        packageMinorVersion="0"
        label="UICC applet"
        cattpPort="105">
        <SecurisationSet>
            <Securisation0348 index="6" spi="1221" kic="00" kid="10"/>
            <Securisation0348 index="5" spi="1200" kic="00" kid="10"/>
            <Securisation0348 index="4" spi="0621" kic="10" kid="10"/>
            <Securisation0348 index="3" spi="0221" kic="00" kid="10"/>
            <Securisation0348 index="2" spi="0021" kic="00" kid="00"/>
            <Securisation0348 index="1" spi="0000" kic="00" kid="00"/>
            <Securisation0348 index="0" spi="0000" kic="00" kid="00"/>
        </SecurisationSet>
    </JavaAppletInstanceProperties>
    <JavaAppletInstanceProperties
        instanceAid="A0000000090001FF33FFFF89C0000000"
        appletAid="A0000000182001010301"
        packageAid="A00000001820010106"
```

```
    packageMajorVersion="1"
    packageMinorVersion="0"
    label="SIM applet"
    cattpPort="106">
    <SecurisationSet>
        <Securisation0348 index="6" spi="1221" kic="00" kid="10"/>
        <Securisation0348 index="5" spi="1200" kic="00" kid="10"/>
        <Securisation0348 index="4" spi="0621" kic="10" kid="10"/>
        <Securisation0348 index="3" spi="0221" kic="00" kid="10"/>
        <Securisation0348 index="2" spi="0021" kic="00" kid="00"/>
        <Securisation0348 index="1" spi="0000" kic="00" kid="00"/>
        <Securisation0348 index="0" spi="0000" kic="00" kid="00"/>
    </SecurisationSet>
</JavaAppletInstanceProperties>
<JavaAppletInstanceProperties
    instanceAid="A0000000871002FF33FFFF8901010100"
    appletAid="A0000000182001010302"
    packageAid="A00000001820010106"
    packageMajorVersion="1"
    packageMinorVersion="0"
    label="USIM applet"
    cattpPort="107">
    <SecurisationSet>
        <Securisation0348 index="6" spi="1221" kic="00" kid="10"/>
        <Securisation0348 index="5" spi="1200" kic="00" kid="10"/>
        <Securisation0348 index="4" spi="0621" kic="10" kid="10"/>
        <Securisation0348 index="3" spi="0221" kic="00" kid="10"/>
        <Securisation0348 index="2" spi="0021" kic="00" kid="00"/>
        <Securisation0348 index="1" spi="0000" kic="00" kid="00"/>
        <Securisation0348 index="0" spi="0000" kic="00" kid="00"/>
    </SecurisationSet>
</JavaAppletInstanceProperties>
<JavaAppletInstanceProperties
    instanceAid="A0000000090002FF33FFFF89B0000100"
        appletAid="A0000000182001030201"
        packageAid="A00000001820010302"
        packageMajorVersion="1"
        packageMinorVersion="0"
        label="UICC interpreter"
        cattpPort="108">
    <SecurisationSet>
        <Securisation0348 index="6" spi="1221" kic="00" kid="10"/>
        <Securisation0348 index="5" spi="1200" kic="00" kid="10"/>
        <Securisation0348 index="4" spi="0621" kic="10" kid="10"/>
        <Securisation0348 index="3" spi="0221" kic="00" kid="10"/>
        <Securisation0348 index="2" spi="0021" kic="00" kid="00"/>
        <Securisation0348 index="1" spi="0000" kic="00" kid="00"/>
        <Securisation0348 index="0" spi="0000" kic="00" kid="00"/>
    </SecurisationSet>
</JavaAppletInstanceProperties>
<JavaAppletInstanceProperties
    instanceAid="A0000000090002FF33FFFF89B0000000"
    appletAid="A0000000182001030201"
    packageAid="A00000001820010302"
    packageMajorVersion="1"
    packageMinorVersion="0"
```



```
<CardFile identifier="2FE2" name="ICCID" type="2"
invalidate="false" fileSize="10"/>
    <CardFile identifier="2F05" name="PL" type="2"
invalidate="false" fileSize="6"/>
</CardFile>
<JavaPackage aid="A000000018200100" majorVersion="1"
minorVersion="0" estimatedSize="0">
    <JavaApplet aid="A00000001820010002">
        <JavaAppletInstance
            aid="A000000018434DFF33FFFF89000000"
            status="PERSONALIZED"
            estimatedSize="0"
            securityDomain="A000000018434DFF33FFFF89000000"
            configuration=""
            tar="000000"
            privileges="8AFE80"
            isIssuerSD="true"
            ownerIdentificationNumber="000000000000"/>
        <JavaAppletInstance aid="A000000018434DFF33FFFF89C00000"
            status="PERSONALIZED"
            estimatedSize="0"
            securityDomain="A000000018434DFF33FFFF89000000"
            configuration=""
            tar="C00000"
            privileges="E00000"
            isIssuerSD="false"
            ownerIdentificationNumber="000000000000"/>
        <JavaAppletInstance aid="A000000018434DFF33FFFF89C00001"
            status="PERSONALIZED"
            estimatedSize="0"
            securityDomain="A000000018434DFF33FFFF89C00000"
            configuration=""
            tar="C00001"
            privileges="800000"
            isIssuerSD="false"
            ownerIdentificationNumber="000000000000"/>
        <JavaAppletInstance aid="A000000018434DFF33FFFF89C00002"
            status="PERSONALIZED"
            estimatedSize="0"
            securityDomain="A000000018434DFF33FFFF89C00002"
            configuration=""
            tar="C00002"
            privileges="804000"
            isIssuerSD="false"
            ownerIdentificationNumber="000000000000"/>
        <JavaAppletInstance aid="A000000018434DFF33FFFF89C00003"
            status="PERSONALIZED"
            estimatedSize="0"
            securityDomain="A000000018434DFF33FFFF89C00003"
            configuration=""
            tar="C00003"
            privileges="A00000"
            isIssuerSD="false"
            ownerIdentificationNumber="000000000000"/>
    </JavaApplet>
</JavaPackage>
```

```
<JavaPackage aid="A00000001820010106" majorVersion="1"
minorVersion="0" estimatedSize="0">
    <JavaApplet aid="A0000000182001010300">
        <JavaAppletInstance
            aid="A0000000871001FF33FFFF8901010100"
            status="SELECTABLE"
            estimatedSize="0"
            configuration=""
            tar="010100"
            privileges="000000"
            isIssuerSD="false"
            ownerIdentificationNumber="0000000000000000"/>
    </JavaApplet>
    <JavaApplet aid="A0000000182001010301">
        <JavaAppletInstance
            aid="A000000090001FF33FFFF89C0000000"
            status="SELECTABLE"
            estimatedSize="0"
            configuration=""
            tar="010101"
            privileges="000000"
            isIssuerSD="false"
            ownerIdentificationNumber="0000000000000000"/>
    </JavaApplet>
    <JavaApplet aid="A0000000182001010302">
        <JavaAppletInstance
            aid="A0000000871002FF33FFFF8901010100"
            status="SELECTABLE"
            estimatedSize="0"
            configuration=""
            tar="010102"
            privileges="000000"
            isIssuerSD="false"
            ownerIdentificationNumber="0000000000000000"/>
    </JavaApplet>
</JavaPackage>
<JavaPackage aid="A00000001820010302" majorVersion="1"
minorVersion="0" estimatedSize="0">
    <JavaApplet aid="A0000000182001030201">
        <JavaAppletInstance
            aid="A000000090002FF33FFFF89B0000100"
            status="SELECTABLE"
            estimatedSize="0"
            configuration=""
            tar="B00001"
            privileges="000000"
            isIssuerSD="false"
            ownerIdentificationNumber="0000000000000000"/>
    <JavaAppletInstance
            aid="A0000000090002FF33FFFF89B0001000"
            status="SELECTABLE"
            estimatedSize="0"
            configuration=""
            tar="B00010"
            privileges="000000"
            isIssuerSD="false"
```

```
        ownerIdentificationNumber="0000000000000000"/>
<JavaAppletInstance
    aid="A0000000090002FF33FFFF89B0000000"
    status="SELECTABLE"
    estimatedSize="0"
    configuration=""
    tar="B00000"
    privileges="000000"
    isIssuerSD="false"
    ownerIdentificationNumber="0000000000000000"/>
</JavaApplet>
</JavaPackage>
</CardStructure>
</CardProfile>
</CardProfileModule>
```

Management Interface Equivalent

Card profiles can be created and updated using the management interface (**Card manager > Card > Profile**). However, not all elements of card profiles can be provisioned in the management interface.

The management interface equivalents of card profile batchload files are:

- 1 The `CardProfile` element and attributes are represented on the **Description** property sheet:

Example 1 - Card Profile Description in the Management Interface

```
<CardProfile
    name="mNFC_v1.0"
    state="1"
    phase="2+"
    cardVendor="Gemplus"
    cardDefinition="mNFC_v1.0"
    serviceImpl=""
    capabilities="0348;CATPP">
    ...

```

Card profile description

Profile name*: mNFC_v1.0
State: Active
Phase*: 2+
Card vendor: Gemplus
Card definition: mNFC_v1.0
OTA Capabilities: ESMSV1 ESMSV2 03.48
CAT-TP compliant: Yes No
Other capabilities:
Creation date: 26 May 2008 11:00:54
Last modification date: 29 May 2008 15:01:51

Service implementation list

Available services	Selected services
2G_0348_Activate_ADMIN 2G_0348_Activate_FDN 2G_0348_ExecScript 2G_0348_Update_ACC 2G_0348_Update_ADMIN	3G_0348_ExecScript 3G_0348_ManageHPLMNwAcT 3G_0348_ManageOPLMNwAcT 3G_0348_ManagePLMNwAcT 3G_0348_UpdateACC

All mandatory fields are specified by a *

Update card profile | **Back**

- 2** The EntityProperties element and the JavaPackageProperties, JavaAppletProperties, and JavaAppletInstanceProperties subelements can only be provisioned by batchloading.

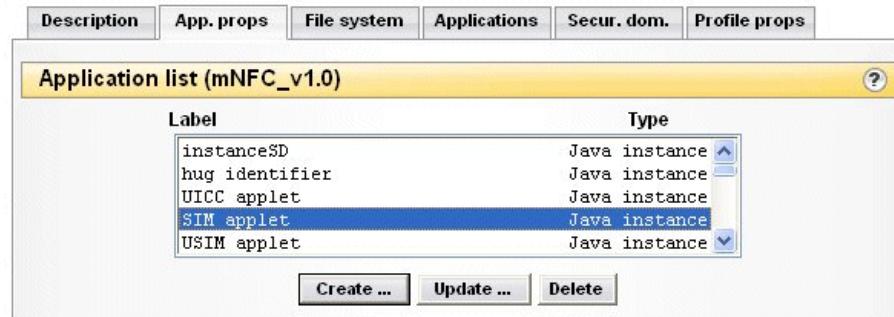
The values of the label attribute are however used to generate the list of applications displayed on the **App. properties** property sheet as the **Application list**.

For example:

Example 2 - The Application List in the Management Interface

```
<EntityProperties>
  <JavaAppletProperties
    appletAid="A0000000182001010300"
    packageAid="A00000001820010106"
    packageMajorVersion="1"
    packageMinorVersion="0"
    label="UICC applet"
    concatenationAvailable="true"
    isSelectable="true"/>
  <JavaAppletProperties
    appletAid="A0000000182001010301"
    packageAid="A00000001820010106"
    packageMajorVersion="1"
    packageMinorVersion="0"
    label="SIM applet"
    concatenationAvailable="true"
    isSelectable="true"/>
```

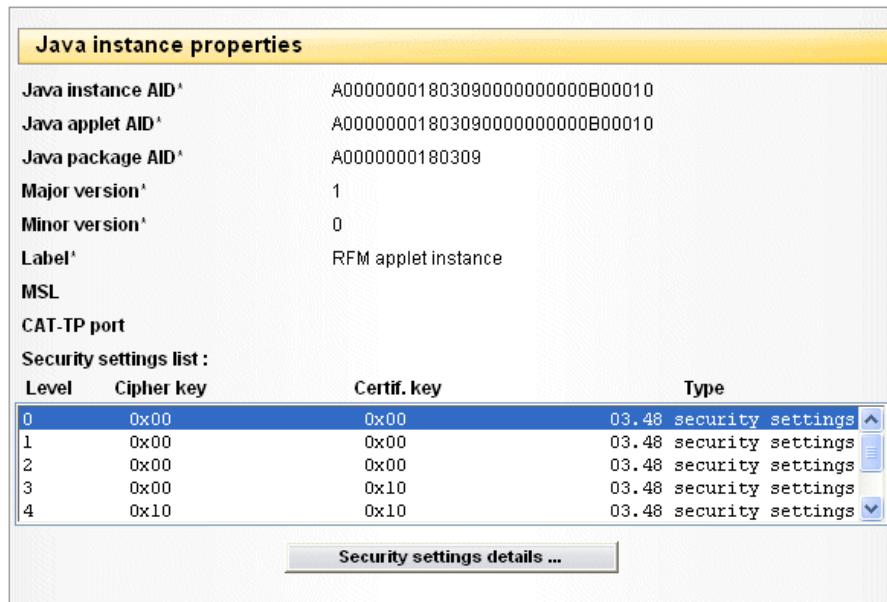
```
<JavaAppletProperties
    appletAid="A0000000182001010302"
    packageAid="A00000001820010106"
    packageMajorVersion="1"
    packageMinorVersion="0"
    label="USIM applet"
    concatenationAvailable="true"
    isSelectable="true"/>
```



- 3 The **JavaAppletInstanceProperties** element and subelements are shown on the **Java instance properties** window. The **SecurisationSet** element and subelements represent the **Security Settings List** area of the window. For example:

Example 3 - Java Instance Properties in the Management Interface

```
<JavaAppletInstanceProperties
    instanceAid="A00000001803090000000000B00010"
    appletAid="A00000001803090000000000B00010"
    packageAid="A0000000180309"
    packageMinorVersion="0" packageMajorVersion="1"
    label="RFM applet instance" >
    <SecurisationSet>
        <Securisation0348 index="4" spi="1E21" kic="10" kid="10"/>
        <Securisation0348 index="3" spi="1621" kic="10" kid="10"/>
        <Securisation0348 index="2" spi="0621" kic="10" kid="10"/>
        <Securisation0348 index="1" spi="0221" kic="00" kid="10"/>
        <Securisation0348 index="0" spi="0000" kic="00" kid="00"/>
    </SecurisationSet>
</JavaAppletInstanceProperties>
```



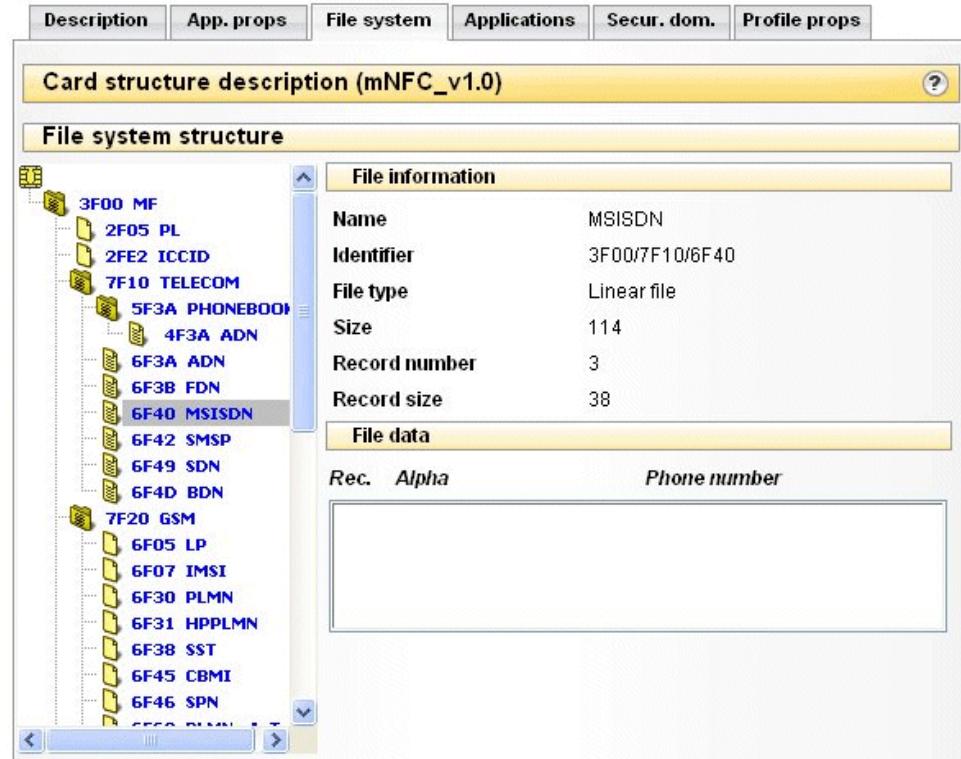
- 4 The CardStructure element and CardFile subelements are represented on the **File System** property sheet of the card profile properties:

Example 4 - File System Structure in the Management Interface

```

<CardStructure name="mNFC_v1.0">
    <CardFile
        identifier="3F00"
        name="MF"
        type="0"
        fileSize="13460">
        <CardFile
            identifier="7F10"
            name="TELECOM"
            type="1"
            fileSize="12378">
            <CardFile
                identifier="5F3A"
                name="PHONEBOOK"
                type="1"
                fileSize="0">
                <CardFile
                    identifier="4F3A"
                    name="ADN"
                    type="3"
                    invalidate="false"
                    recordNumber="0"
                    recordSize="0"/>
            </CardFile>
        </CardFile>
    </CardFile>
    ...
</CardStructure>

```



- 5 The JavaPackage, JavaApplet and JavaAppletInstance subelements of the CardStructure element are represented on the **Applications** property sheet of the card profile properties:

Example 5 - Applications Structure in the Management Interface

```

<CardStructure>
...
<JavaPackage
    aid="A000000018200100"
    majorVersion="1"
    minorVersion="0"
    estimatedSize="0">
    <JavaApplet aid="A0000000182001002">
        <JavaAppletInstance aid="A000000018434DFF33FFFF89000000"
            status="PERSONALIZED"
            estimatedSize="0"
            securityDomain="A000000018434DFF33FFFF89000000"
            configuration=""
            tar="000000"
            privileges="8AFE80"
            isIssuerSD="true"
            ownerIdentificationNumber="000000000000"/>
    </JavaApplet>
</JavaPackage>
...
</CardStructure>

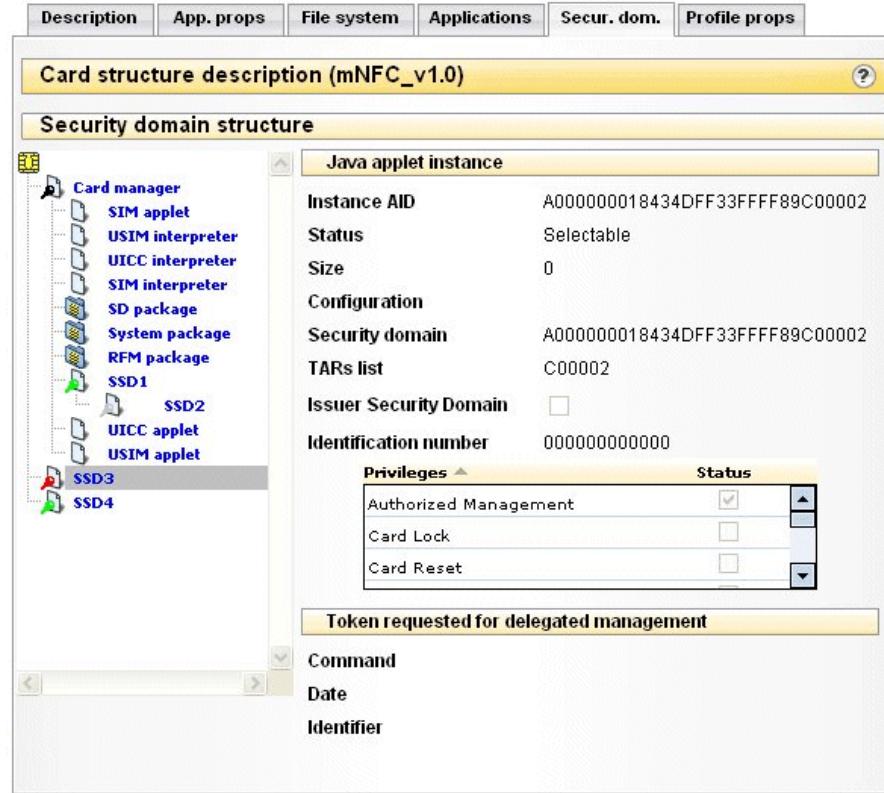
```



- 6 The properties of security domains are displayed on the **Secur. dom.** property sheet of the Card Profile properties window:

Example 6 - Security Domain Properties

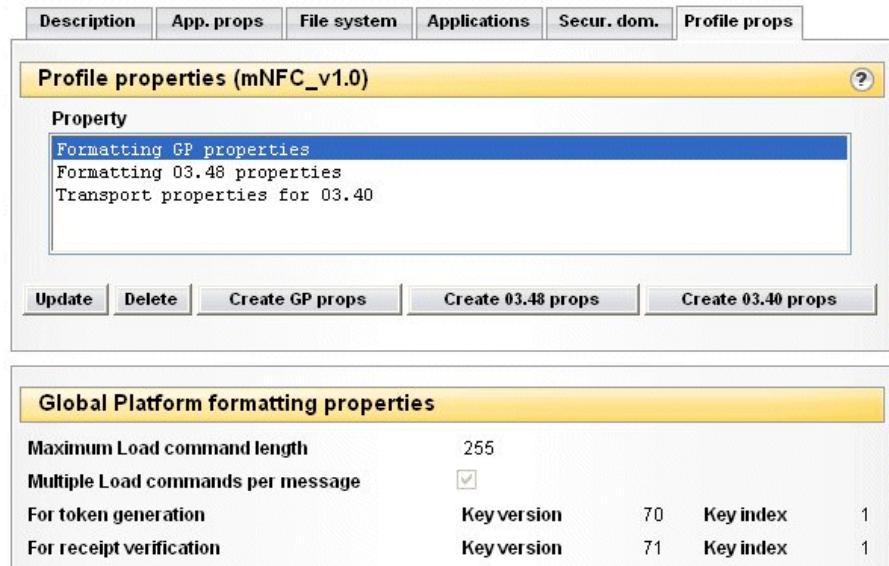
```
<JavaAppletInstance
    aid="A000000018434DFF33FFFF89000000"
    status="PERSONALIZED"
    estimatedSize="0"
    securityDomain="A000000018434DFF33FFFF89000000"
    configuration=""
    tar="000000"
    privileges="8AFE80"
    isIssuerSD="true"
    ownerIdentificationNumber="000000000000"/>
```



- 7 The **FormattingGPProperties** element is shown when selecting **Formatting GP properties** on the **Profile properties** property sheet of the Card Profile properties window. For example:

Example 7 - GP Formatting Properties in the Management Interface

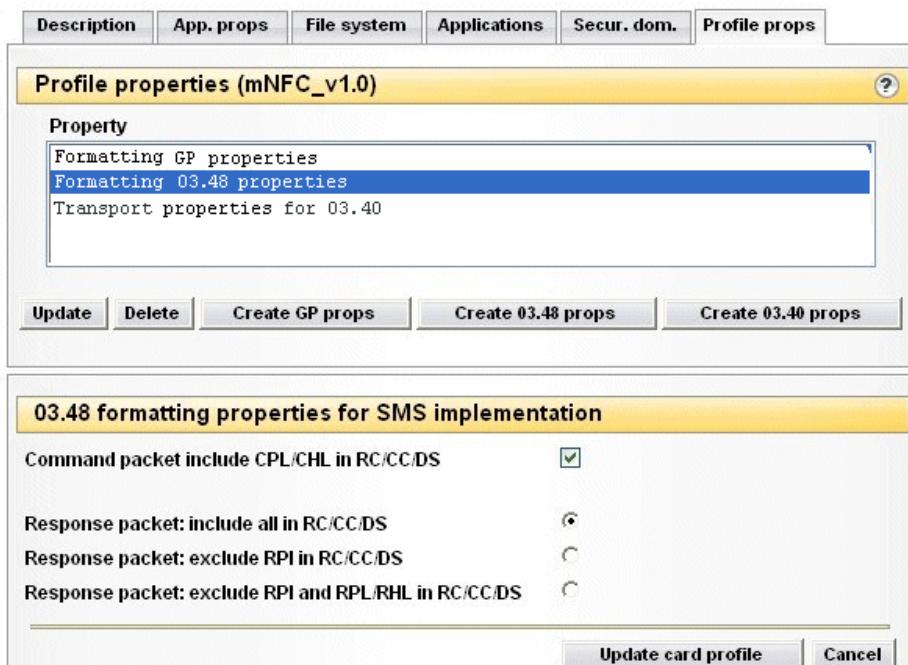
```
<FormattingProperties>
  <FormattingGPProperties
    multipleLoadCommandPerMessage="true"
    maxLoadCommandLength="255"
    tokenKeyVersion="70"
    tokenKeyIndex="1"
    receiptKeyVersion="71"
    receiptKeyIndex="1"/>
</FormattingProperties>
```



- 8 The `Formatting0348Properties` element is shown when selecting **Formatting 03.48 properties** on the **Profile properties** property sheet of the Card Profile properties window. For example:

Example 8 - 03.48 Formatting Properties in the Management Interface

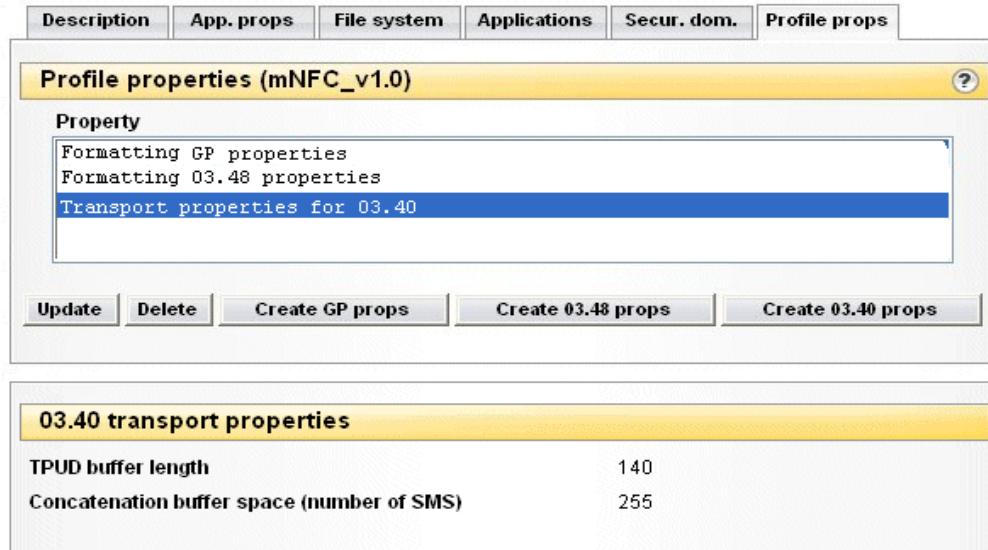
```
<FormattingProperties>
  <Formatting0348Properties
    isCplChlRequiredInCertificate="true"
    rpFieldRequiredInCertificate="RP_CERTIF_ALL"/>
</FormattingProperties>
```



- 9 The **TransportProperties** element and subelements are shown when selecting **Transport properties for 03.40** on the **Profile properties** property sheet of the Card Profile properties. For example:

Example 9 - 03.40 Transport Properties in the Management Interface

```
<TransportProperties>
  <TransportProperties0340
    tpudBufferSize="140"
    concatBufferSpace="255"/>
</TransportProperties>
```



Service Declarations and Implementations

The DTD structure used to represent a service declaration consists of the following parameters:

Table 16 - Service Declaration Parameters

Parameter	Description
name	Name of the service (30 characters maximum).
handlerName	Name of the Java class that manages the service; for example, "rca.serviceDecl.Download".
state	State of the service. Valid values: "ACTIVE" or "INACTIVE".
isCardIndependent	Specifies that the service is card and profile independent or not. Valid values: true, false.

The DTD data structures used to represent a service implementation consist of the following parameters.

Table 17 - Service Implementation Parameters

Parameter	Description
name	Name of the service implementation; for example, "2G-0348-ActivateADN". Assumed to be a unique key (30 characters maximum). This name must be used in the card profile to reference the service implementation.
serviceName	Name of the service declaration.
applicationIdentifier	A reference to an application that contains for a 03.48 application, the Toolkit Application Reference (TAR) number. Can be an application identifier in the case of an ESMS V2 service, or a DF identifier for an ESMS V1 service. 30 characters maximum.
javaClassName	Name of the Java class (if any) that manages the service (100 characters maximum).
scriptName	Full path and file name that defines a script or a template file, if any (100 characters maximum).
state	State of the service implementation. Valid values: "0" (Inactive) or "1" (Active).
securityLevel	Security level used to communicate with the application.
cardContentUpdateActivated	Whether to update the contents of card instances targeted by service implementations upon successful completion.
cardContentCheckActivated	Whether to check the contents of card instances targeted by service implementations.
transportInformation	PID values and DCS value.

DTD Structure

```

<!-- Definition of Card Service Implementation Module -->
<!ELEMENT ServiceImplModule (ServiceImpl)+>
<!ELEMENT ServiceImpl (SMS03_40?)>
<!ATTLIST ServiceImpl name CDATA #REQUIRED>
<!ATTLIST ServiceImpl serviceName CDATA #REQUIRED>
<!ATTLIST ServiceImpl javaClassName CDATA #IMPLIED>
<!ATTLIST ServiceImpl scriptName CDATA #IMPLIED>
<!ATTLIST ServiceImpl state CDATA #IMPLIED>
<!ATTLIST ServiceImpl
    cardContentUpdateActivated (true|false) #REQUIRED>
<!ATTLIST ServiceImpl cardContentCheckActivated (true|false) #IMPLIED>
<!ATTLIST ServiceImpl applicationIdentifier CDATA #IMPLIED>
<!ATTLIST ServiceImpl securityLevel CDATA #IMPLIED>
<!ELEMENT SMS03_40 EMPTY>
<!ATTLIST SMS03_40 pid CDATA #REQUIRED>
<!ATTLIST SMS03_40 dcs CDATA #REQUIRED>

```

Example

The following example shows how to prepare an XML file for a service implementation:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ServiceImplModule SYSTEM "cardframework.dtd">

<ServiceImplModule>
    <ServiceImpl
        name = "3G_0348_ExecScript"
        serviceName = "ExecScript_3G"
        state = "1"
        securityLevel="0"
        applicationIdentifier="B00001"
        javaClassName = "rca.serviceImpl.generic0348.ExecScript"
        cardContentUpdateActivated = "false"
        cardContentCheckActivated = "true"
        scriptName = ""
    />
    <ServiceImpl
        name = "3G_0348_UpdateSPN"
        serviceName = "UpdateSPN_3G"
        state = "1"
        securityLevel="0"
        applicationIdentifier="B00001"
        cardContentUpdateActivated = "false"
        cardContentCheckActivated = "true"
        scriptName = "rca/serviceImpl/generic0348/UpdateSPN_3G.script"
        javaClassName = "rca.serviceImpl.generic0348.UpdateSPN_3G"
    />
    <ServiceImpl
        name = "3G_0348_UpdateUST"
        serviceName = "UpdateUST_3G"
        state = "1"
        securityLevel="0"
        applicationIdentifier="B00001"
        cardContentUpdateActivated = "false"
        cardContentCheckActivated = "true"
        scriptName = "rca/serviceImpl/generic0348/UpdateUST_3G.script"
        javaClassName = "rca.serviceImpl.generic0348.UpdateUST_3G"
    />
</ServiceImplModule>
```

Management Interface Equivalent

Service implementation definitions can be created and updated using the management interface (**Card manager > Service > Create**).

The management interface equivalents of service implementation definition are shown in “Figure 66”:

Figure 66 - Service Implementation in the Management Interface

```
<ServiceImplModule>
    <ServiceImpl
        name = "ActivateADN_For_GemXplore_98"
        serviceName = "2G_0348_Activate_ADN"
        state = "1"
```

```

    securityLevel="0"
    applicationIdentifier="000001"
    cardContentUpdateActivated = "false"
    scriptName = "rca/serviceImpl/gemplus/ActivateADN"
    javaClassName = "rca.serviceImpl.gemplus.ActivateADN"
/>

```

Service implementation information

Service name	2G_0348_Activate_ADN
State	Active
Java implementation class*	rca.serviceImpl.generic0348.Ac
Script template	rca/serviceImpl/generic0348/Ac
Security level*	0
Application target identifier	B00010
Card content checked	<input checked="" type="checkbox"/>
Card content updated	<input type="checkbox"/>
Transport	▼

All mandatory fields are specified by a *

Validate **Cancel**

Card Instances

Card instances describe all the data present on a card that is of interest for OTA management of the card.

When provisioning a card, a reference to the card profile it is based on is mandatory. By default, the card content (in terms of files, Java applets and packages, and native applications) is the same as that of the card profile (although some elements can be identified as differing from those of the card profile by including them in the **SIMCard** element).

Table 18 - Card Content Parameters

Parameter	Description
serialNumber	Mandatory. Unique identifier of a card (also known as the ICCID) in the repository (10 bytes BCD-coded, giving 20 digits maximum). This must be fully digitally coded.
profileName	Mandatory. Reference to a card profile (120 ASCII characters or 30 UTF-8 characters maximum).
state	State of the card. Valid values: "0" (Inactive) or "1" (Active).
imsi	International Mobile Subscriber Identity - subscriber's IMSI number (8 bytes BCD-coded, giving 15 digits maximum). This IMSI number must be unique in the Card Manager database and must be fully digitally coded.
msisdn	The Mobile Station International Subscriber Directory Number (MSISDN) - the telephone number associated with the SIM card and subscriber (10 BCD-coded bytes, giving 20 digits maximum). This MSISDN number must be unique in the Card Manager database and must be fully digitally coded.
linkedCard	Attribute that can be used by an external application to manage dual cards.

Table 18 - Card Content Parameters (continued)

Parameter	Description
freeEEPROMSize	Optional. The amount of space (in bytes) free in the EEPROM memory.
freeVolatileSize	Optional. The amount of free volatile memory (in bytes).
freeNonVolatileSize	Optional. The amount of free non volatile memory (in bytes).
checkSecurity	Optional. If set to true, the platform checks when provisioning the card that the security keys have already been provisioned for this card. If the keys are present, the card instance is created. If not, the card is not provisioned. If set to false or not present, the security keys' presence is not checked and the card instance is always provisioned.
CardFile, JavaPackage, JavaAppletInstance	Elements used to indicate specialized content for the current card.

Note: The **iccid**, **imsi**, and **msisdn** elements must be fully digitally coded. A check rejects provisioning of a card if any of these IDs contain a non-decimal value (0-9).

DTD Structure

```
<!-- Definition of the Card Content and Smart Cards Module -->
<!ELEMENT CardContentModule (SmartCard*, SimCard*)>
<!ATTLIST CardContentModule profileName CDATA #IMPLIED>
<!ELEMENT SmartCard (CardFile*)>
<!ATTLIST SmartCard serialNumber CDATA #REQUIRED>
<!ATTLIST SmartCard profileName CDATA #IMPLIED>
<!ATTLIST SmartCard state CDATA #IMPLIED>

<!ELEMENT SimCard (CardFile*, JavaPackage*, JavaAppletInstance*)>
<!ATTLIST SimCard serialNumber CDATA #REQUIRED>
<!ATTLIST SimCard profileName CDATA #IMPLIED>
<!ATTLIST SimCard state CDATA #IMPLIED>
<!ATTLIST SimCard imsi CDATA #IMPLIED>
<!ATTLIST SimCard msisdn CDATA #IMPLIED>
<!ATTLIST SimCard linkedcard CDATA #IMPLIED>
<!ATTLIST SimCard freeEEPROMSize CDATA #IMPLIED>
<!ATTLIST SimCard freeVolatileSize CDATA #IMPLIED>
<!ATTLIST SimCard freeNonVolatileSize CDATA #IMPLIED>
<!ATTLIST SimCard checkSecurity (true|false) #IMPLIED>
```

Example

The following example shows how to prepare an XML file to create card instances without an MSISDN:

```
<!DOCTYPE CardContentModule SYSTEM "cardframework.dtd">
<CardContentModule>
  <SimCard serialNumber="100010099988812345"
    imsi="123010099965432" profileName="GX98_16K" state="1"
    checkSecurity="true"/>
  <SimCard serialNumber="100010099988812346"
    imsi="123010099965433" profileName="GX98_16K" state="1"
    checkSecurity="true"/>
```

```

<SimCard serialNumber="100010099988812347"
    imsi="123010099965434" profileName="GX98_16K" state="1"
    checkSecurity="true"/>
<SimCard serialNumber="100010099988812348"
    imsi="123010099965435" profileName="GX98_16K" state="1"
    checkSecurity="true"/>
<SimCard serialNumber="100010099988812349"
    imsi="123010099965436" profileName="GX98_16K" state="1"
    checkSecurity="true"/>
</CardContentModule>

```

The following example shows how to prepare an XML file to create card instances with an MSISDN:

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CardContentModule SYSTEM "cardframework.dtd">
<CardContentModule>
    <SimCard serialNumber="100010099988812345"
        imsi="123010099965432" msisdn="0601000100"
        profileName="GX98_16K" state="1" checkSecurity="true"/>
    <SimCard serialNumber="100010099988812346"
        imsi="123010099965433" msisdn="0601000101"
        profileName="GX98_16K" state="1" checkSecurity="true"/>
    <SimCard serialNumber="100010099988812347"
        imsi="123010099965434" msisdn="0601000102"
        profileName="GX98_16K" state="1" checkSecurity="true"/>
    <SimCard serialNumber="100010099988812348"
        imsi="123010099965435" msisdn="0601000103"
        profileName="GX98_16K" state="1" checkSecurity="true"/>
    <SimCard serialNumber="100010099988812349"
        imsi="123010099965436" msisdn="0601000104"
        profileName="GX98_16K" state="1" checkSecurity="true"/>
</CardContentModule>

```

The following example shows how to prepare an XML file to create a card content entry for a SIM card. It creates a card whose ICCID is “100010099988812345” and which contains a Java package (AID=“A0000000180310”, major version “1”, minor version “0”) consisting of a single Java applet and a single applet instance:

```

<!DOCTYPE CardStructureModule SYSTEM "cardframework.dtd">
<CardContentModule>
    <SimCard serialNumber="100010099988812345" imsi="123010099965432"
        msisdn="0600000000" profileName="GXX-V3.2_128K" state="1"
        checkSecurity="true">

    <!-- Example applet : no real values -->
    <JavaPackage
        aid="A0000000180310" majorVersion="1" minorVersion="0"
        securityDomain="A0000000018434D08090A0B0C000000"
        estimatedSize="0"
        <JavaApplet aid="A000000001803010000000000B00010"
            <JavaAppletInstance aid="A000000001803010000000000B00011"
            status="SELECTABLE" estimatedSize="0"
            configuration=""
            securityDomain="A0000000018434D08090A0B0C000000"
            tar="B000010"/>
        </JavaApplet>
    </JavaPackage>
</SimCard>

```

```
</CardContentModule>
```

The following example shows how to prepare an XML file to update card instances with an MSISDN (the MSISDN being allocated after the card instance creation) and simultaneously activate the card:

```
<!DOCTYPE CardContentModule SYSTEM "cardframework.dtd">
<CardContentModule>
  <SimCard imsi="123010099965432" msisdn="0601000100" state="1"/>
  <SimCard imsi="123010099965433" msisdn="0601000101" state="1"/>
  <SimCard imsi="123010099965434" msisdn="0601000102" state="1"/>
  <SimCard imsi="123010099965435" msisdn="0601000103" state="1"/>
  <SimCard imsi="123010099965436" msisdn="0601000104" state="1"/>
</CardContentModule>
```

Card Security

The card security DTD is used to represent security settings for batchloading into the Card Security database.

Card-specific security data is enclosed within an **MBCard** element. Each **MBCard** element represents a single card stored in the database and the associated security parameters that apply to the card. The security parameters themselves are defined in a sub-element of **MBCard** called either **SecurityData** or **SecurityDomain**:

- The **SecurityData** element implements the Security Data security model and is used to provision native card security. This is the same representation as that provided for GemXplore Suite V2 SP4 and earlier versions of the product, and existing security files can be provisioned without any modifications. The **SecurityData** model was also used to provision GSM 03.48 security information but with one limitation: it was impossible to specify a synchronization counter for each key set. Use of the **SecurityData** model is now deprecated.
- The **SecurityDomain** element implements the Security Domain security model. It allows you to specify:
 - One synchronization counter for each key set
 - A default implicit algorithm number for redundancy checks
 - A default proprietary algorithm number for redundancy checks.

The following are attributes of the **MBCard** element:

Table 19 - MBCard Element Parameters

Parameter	Description
serialNumber	Identifies the serial number of the card to which the security settings are linked.

Table 19 - MBCard Element Parameters

Parameter	Description
transportKey	Deprecated. You should only define a single transport key for the whole file: define this attribute on the MBCardModule element
algoName	Text identifying the algorithm to use to encrypt the key values: "DES", "TripleDES", "DES_ECB", "TripleDES_ECB", "XOR". If omitted, "DES" is used by default.
checkKey	Optional. Present only if the transportKey and algoName attributes are present, otherwise ignored. If present, the checkKey attribute must contain the reference hexadecimal value "00112233445566778899AABBCCDDEEFF" ciphered with the transport key (transportKey) and specified algorithm (algoName). The platform deciphers this checkKey . If the reference value is found, the cards are provisioned. Otherwise, the provisioning file is rejected and cards are not provisioned.

The following are attributes of the **SecurityData** element:

Table 20 - SecurityData Element Parameters

Parameter	Description
securityID	Reference to the key container file on the card.
keysID	Key set identifier. Not used for native applications and its value must be set to "1".
algoNumber	Identifies the algorithm to be used. An integer value identifying one of the algorithms supported by the Library V3 formatting library.
sync	Synchronization counter value on the card. There is normally only one synchronization counter on a native card, coded on 5 bytes.
syncID	Optional reference to a card file containing a synchronization counter.

The following are attributes of the **KeyValue** element that represents an individual key value within a **SecurityData** key set:

Table 21 - KeyValue Element Parameters

Parameter	Description
recordNb	Integer identifying the key number within the key file.
value	The value of the key. The key length depends on the algorithm used, and is checked for compatibility with the algorithm during batchloading.

The following are attributes of the **SecurityDomain** element:

Table 22 - SecurityElement Parameters

Parameter	Description
securityDomainID	The AID of the on-card security domain entity responsible for managing keys and security computations.
defaultSync	Default synchronization counter value of the security domain (see notes below).
defaultSyncID	Reference to the default synchronization counter of the security domain (see notes below).
implicitRcAlgoNumber	Default implicit algorithm for redundancy check (RC) computation.
proprietaryRcAlgoNumber	Default proprietary algorithm for RC computation.
transportKey	Deprecated. You should only define one transport key for the entire file: define it on the MBCardModule element.
algoName	Optional, but if defined specifies the algorithm to use to decipher the key value of the security domain.

Specifying different combinations of the **defaultSync** and **defaultSyncID** attributes produces:

- **defaultSync** defined and **defaultSyncID** not defined: the platform creates a synchronization counter with the same identifier as that of the security domain and the value specified with the **defaultSync** attribute. The security domain references this counter by default.
- **defaultSync** not defined and **defaultSyncID** defined: the platform does not create a synchronization counter and the security domain references the default counter identifier par default (the counter itself must have been defined earlier in the batchload file).
- **defaultSync** defined and **defaultSyncID** defined. The platform creates a synchronization counter with the identifier and value specified, and the security domain references this counter by default.
- Neither **defaultSync** nor **defaultSyncID** defined. Produces a batchloading error.

The following are attributes and sub-elements of the **KeySet** element that represents an individual key set within a **SecurityDomain** key set:

Table 23 - KeySet Element Parameters

Parameter	Description
versionNumber	The number of the key set, from 1 to 15.
sync	Value of the key set's synchronization counter (see notes below).
syncId	Reference to the synchronization counter used by the key set (see notes below).
Kic	The key set's KiC key. At least one of the Kic and Kid elements must be present.
Kid	The key set's KID key. At least one of the Kic and Kid elements must be present.
Kik	The key set's KiK key. Optional.
value	The KiC, KID, or KiK key value. The key length depends on the algorithm used, and is checked for compatibility with the algorithm during batchloading.
algoNumber	Integer that identifies the algorithm to use. Refer to "Table 24 - Security Algorithms".

Specifying different combinations of the **sync** and **syncID** attributes produces:

- **sync** defined and **syncID** not defined: the platform creates a synchronization counter with the same identifier as that of the security domain and the value specified with the **sync** attribute. The security domain references this counter by default.
- **sync** not defined and **syncID** defined: the platform does not create a synchronization counter and the security domain references the default counter identifier par default (the counter itself must have been defined earlier in the batchload file).
- **sync** defined and **syncID** defined. The platform creates a synchronization counter with the identifier and value specified, and the security domain references this counter by default.
- Neither **sync** nor **syncID** defined. Accepted; no synchronization counter is specified for the key set.

Table 24 - Security Algorithms

algoNumber	Algorithm	Value	Remarks
0	XOR8	8	CC
1	COMP128	16	CC
2	DES CBC/ECB (use mode requested in KiC byte)	8	CC, Ciphering
3	3-DES CBC	16/24	CC, Ciphering
4	COMP128V2	16	CC
5	CRC-32		RC
6	XOR4		RC

Table 24 - Security Algorithms

algoNumber	Algorithm	Value	Remarks
10	DES/CBC/None	8	Ciphering
11	DES/ECB/None	8	Ciphering
12	3-DES/CBC/None	16/24	Ciphering
13	3-DES/ECB/None	16/24	Ciphering
30	CRC-16 A		RC
128	DES/CBC/NOPADDING	8	
129	3-DES/CBC/NOPADDING	16/24	
130	3-DES/CBC/NOPADDING	16/24	
131	DES/ECB/NOPADDING	8	
132	DES/ECB/NOPADDING	8	
160	RSA	512/1024	Public exponent e component (clear text)
161	RSA	512/1024	Modulus n component (clear text)
162	RSA	512/1024	Modulus n component
163	RSA	512/1024	Private exponent d component
164	RSA	512/1024	Chinese remainder p component
165	RSA	512/1024	Chinese remainder q component
166	RSA	512/1024	Chinese remainder pq component
167	RSA	512/1024	Chinese remainder dp1component
168	RSA	512/1024	Chinese remainder dq1component

DTD Structure

```

<?xml version="1.0" encoding="UTF-8"?>

<!-- Definition of Message Builder Module --&gt;
&lt;!ELEMENT MBCardModule (MBCard)+&gt;
&lt;!ATTLIST MBCardModule transportKey CDATA#IMPLIED
      algoName (DES|TripleDES|DES_ECB|TripleDES_ECB|XOR) #IMPLIED
      checkKey CDATA #IMPLIED&gt;

<!-- Definition of SmartCard --&gt;
&lt;!ELEMENT MBCard (SecurityData*, SecurityDomain*)&gt;
&lt;!ATTLIST MBCard serialNumber CDATA#REQUIRED
      transportKey CDATA#IMPLIED
</pre>

```

```

algoName (DES|TripleDES|DES_ECB|TripleDES_ECB|XOR) # IMPLIED>

<!-- Definition of SecurityData -->
<!ELEMENT SecurityData (Sync?, Keys?)>
<!ATTLIST SecurityData securityIDCDATA#REQUIRED
  keysIDCDATA#REQUIRED
  sync CDATA # IMPLIED
  syncIDCDATA#IMPLIED>

<!-- Definition of Sync -->
<!ELEMENT Sync EMPTY>
<!ATTLIST Sync valueCDATA#REQUIRED>

<!-- Definition of Keys -->
<!ELEMENT Keys (KeyValue)+>

<!-- Definition of KeyValue -->
<!ELEMENT KeyValue EMPTY>
<!ATTLIST KeyValue recordNbCDATA#REQUIRED
  value CDATA # REQUIRED
  algoNumberCDATA#REQUIRED>

<!-- Definition of SecurityDomain -->
<!ELEMENT SecurityDomain (Sync?, Keyset+)>
<!ATTLIST SecurityDomain domainIDCDATA#REQUIRED
  defaultSyncCDATA#IMPLIED
  defaultSyncIDCDATA#IMPLIED
  implicitRcAlgoNumberCDATA#REQUIRED
  proprietaryRcAlgoNumberCDATA#IMPLIED
  transportKeyCDATA#IMPLIED
  algoName (DES|TripleDES|DES_ECB|TripleDES_ECB|XOR) # IMPLIED>

<!-- Definition of Keyset -->
<!ELEMENT Keyset (Sync?, Kic?, Kid?, Kik?, Key*)>
<!ATTLIST Keyset versionNumberCDATA#REQUIRED
  sync CDATA # IMPLIED
  syncIDCDATA#IMPLIED>

<!-- Definition of Key -->
<!-- index corresponds to key index as defined in GP -->
<!-- type corresponds to key type defined in GP 11.1.8 -->
<!ELEMENT Key (KeyComponent+)>
<!ATTLIST Key index (1|2|3) #REQUIRED>
<!ELEMENT KeyComponent EMPTY>
<!ATTLIST KeyComponent type CDATA #REQUIRED
  value CDATA #REQUIRED>

<!-- Definition of Kic -->
<!ELEMENT Kic EMPTY>
<!ATTLIST Kic valueCDATA#REQUIRED
  algoNumberCDATA#REQUIRED>

<!-- Definition of Kid -->
<!ELEMENT Kid EMPTY>
<!ATTLIST Kid valueCDATA#REQUIRED

```

```

algoNumberCDATA#REQUIRED>

<!-- Definition of Kik -->
<!ELEMENT Kik EMPTY>
<!ATTLIST KikvalueCDATA#REQUIRED
    algoNumberCDATA#REQUIRED>

```

Examples

The following is an example of an XML file used for security key provisioning, using one synchronization counter for each key set and a default synchronization counter for the security domain:

```

<MBCard serialNumber="100010099988812345">
    <SecurityDomain securityDomainID="A000000018434D08090A0B0C000000">
        defaultSyncID="default" implicitRcAlgoNumber="6"
        proprietaryRcAlgoNumber="6">
        <Sync value="0000000000"/>
        <Keyset versionNumber="1" syncID="Domain1KeySet1">
            <Sync value="0000000001"/>
            <Kic value="000102030405060708090A0B0C0D0E0F"
                algoNumber="12"/> <!-- 3DES/CBC -->
            <Kid value="000102030405060708090A0B0C0D0E0F"
                algoNumber="3"/> <!-- 3DES -->
        </Keyset>
        <Keyset versionNumber="2" syncID="Domain1KeySet2" >
            <Sync value="0000000001"/>
            <Kic value="000102030405060708090A0B0C0D0E0F"
                algoNumber="12"/> <!-- 3DES/CBC -->
            <Kid value="000102030405060708090A0B0C0D0E0F"
                algoNumber="3"/> <!-- 3DES -->
        </Keyset>
    </SecurityDomain>
</MBCard>

```

The following example shows the same synchronization counter being used by all key sets:

```

<MBCard serialNumber="2000003" >
    <SecurityDomain securityDomainID="A000000018434D08090A0B0C0D0000">
        defaultSyncID="default3" implicitRcAlgoNumber="5"
        proprietaryRcAlgoNumber="5">
        <Sync value="0000000001"/>
        <Keyset versionNumber="1" syncID="default3" >
            <Kic value="000102030405060708090A0B0C0D0E0F"
                algoNumber="12"/> <!-- 3DES/CBC -->
            <Kid value="000102030405060708090A0B0C0D0E0F"
                algoNumber="3"/> <!-- 3DES -->
        </Keyset>
        <Keyset versionNumber="3" syncID="default3" >
            <Kic value="000102030405060708090A0B0C0D0E0F"
                algoNumber="12"/> <!-- 3DES/CBC -->
            <Kid value="000102030405060708090A0B0C0D0E0F"
                algoNumber="3"/> <!-- 3DES -->
        </Keyset>
    </SecurityDomain>

```

```
</MBCard>
```

Group Files

The XML format uses the following DTD:

```
<!ELEMENT CardGroupList (CardGroup)*>
<!ELEMENT CardGroup (Card)*>
<!ATTLIST CardGroup id CDATA #REQUIRED>
<!ATTLIST CardGroup description CDATA #REQUIRED>

<!ELEMENT Card EMPTY>
<!ATTLIST Card iccid CDATA #IMPLIED>
<!ATTLIST Card msisdn CDATA #IMPLIED>
<!ATTLIST Card imsi CDATA #IMPLIED>
```

Example

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--DOCTYPE file SYSTEM "volume.dtd"-->
<CardGroupList>
    <CardGroup id="3G_Group_users" description="Users of 3G cards">
        <Card imsi="208200509003041"/>
        <Card msisdn="07613630010501"/>
        <Card iccid="890950010000000917"/>
    </CardGroup>
</CardGroupList>
```

Management Interface Equivalent

Group definitions can be created and updated using the management interface (**Card manager** > **Group management**).

The management interface equivalent of group definition batchloading is shown in “Figure 67”:

Figure 67 - Defining Group Definitions in the Management Interface

```
<CardGroupList>
    <CardGroup id="3GUsers" description="Subscribers with 3G handsets">
    </CardGroup>
</CardGroupList>
```

Group description	
Group ID	3GUsers
Description	Subscribers with 3G handsets

Subscriber Files

Example

The required format of the XML file is shown by the following example. The subscriber **id** is the MSISDN and the **cardId** is the ICCID. The **creatorId** is the name of the user that created the file.

Note that this example shows three subscriber records, the first two with internet access and the third one without.

```
<subsdef>
  <subscriber id="061110000" cardId="48484111564154185">
    <account>
      <param name="creatorId" value="ccagemplus"/>
      <param name="fullName" value="userLamda"/>
      <param name="authMethod" value="mobile"/>
      <param name="status" value="false"/>
      <param name="profileId" value="Subscriber"/>
    </account>
  </subscriber>
  <subscriber id="061100000" cardId="48484111564154186">
    <account>
      <param name="creatorId" value="ccagemplus"/>
      <param name="fullName" value="userLamda"/>
      <!-- expiration date (in milliseconds) =
          (date(01/01/1970)
          - date(15/10/2005))*24*3600*1000 -->
      <param name="expireDate" value="1033377710673"/>
      <param name="authMethod" value="password"/>
      <param name="authValue" value="gemplus13"/>
      <param name="status" value="true"/>
      <param name="profileId" value="subscriberProfile"/>
      <param name="subscriberRef" value="061100000"/>
    </account>
  </subscriber>
  <subscriber id="061200000" cardId="48484111564154187">
  </subscriber>
</subsdef>
```

Management Interface Equivalent

Subscriber definitions can be created and updated using the management interface (**User management > Account > Subscriber > Create new subscriber**).

The management interface equivalent of subscriber imports with web access is shown in "Figure 68":

Figure 68 - Defining Subscriber Accounts with Web Access

```
<subscriber id="0602030405" cardId="48484111564154185">
  <account>
    <param name="creatorId" value="ccagempplus"/>
    <param name="fullName" value="user LAMBDA"/>
    <param name="authMethod" value="mobile"/>
    <param name="status" value="false"/>
    <param name="profileId" value="Subscriber"/>
  </account>
</subscriber>
```

The management interface equivalent of subscriber imports without web access are shown in "Figure 66":

Figure 69 - Defining Subscriber Accounts without Web Access

```
</subscriber>
<subscriber id="0602030405" cardId="5754532873223056875">
</subscriber>
```

Classic Campaign Target Files

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Campaign SYSTEM "campaign.dtd">
<Campaign>
    <CardInformation MSISDN = "97264677667"/>
    <CardInformation MSISDN = "97252499911"/>
</Campaign>
```

XCT Campaign Target Files

XCT campaign target files are plain text files containing lists of subscribers targeted by an XCT campaign.

Target files must have a “.target” extension.

When you upload a targets file to the server, a “.info” file is created automatically by the platform. This file simply contains the number of targets contained in the target file.

Note: If you generate the target file automatically and upload the file directly to the server, you must create the “.info” file yourself.

You manage target files on the Create a New Campaign - Target Set Selection window (**Campaign Management > Targets**).

Example

The following shows an example XCT campaign target file.

Note: “SIMCARD” and “MSISDN” are the only values allowed on the TARGET_TYPE and TARGET_ID lines.

```
TARGET_TYPE=SIMCARD
TARGET_ID_TYPE=MSISDN
0602000000
0602000001
0602000002
0602000003
0602000004
0602000005
0602000006
0602000007
0602000008
0602000009
0602000010
```

Product Parameters

The tables in this chapter describe the program parameters for each product and give the default value of each parameter.

Changing Product Parameter Values

When a OTA Manager V5.1 product instance is created, the default product parameter values are stored in the Framework database.

Thereafter, product parameter values should only be changed through the management interface, as described below.

Caution: Do not attempt to change product parameter values by directly editing the installed data files—change values only through the management interface of product instances.

Modified parameter values are stored in the database and used for all future operations.

When you change a parameter value, it is only changed for the current product instance. You can use the export/import facility (see “Exporting and Importing Product Parameters” on page 172) to copy product parameter values to other product instances.

The product parameter lists in this chapter indicate when the new parameter value is taken into account:

- **H** = “Hot”. Effective immediately after the change
- **C** = “Cold”. Effective only after the next start of the product.

To change the parameter values for a product, for example, the name of an instance of the Card Manager, use the management interface:

- 1 Do either of the following:
 - On the welcome window, click **Platform management**.
 - Click **Platform** in any OTA Manager V5.1 management interface window.
- 2 On the left-hand menu of the Platform management window, click **Product configuration**.
- 3 In the **Product list** window, select a product and click **Configure**.
- 4 Click **Edit Parameters**.

Exporting and Importing Product Parameters

To enable parameter values to be updated easily for several instances of a product, the management interface offers an export/import facility. After making the required changes for one instance, click **Export** to export the values to a file. Then go to the Product Parameters List window for a different instance and click **Import** to import the values from the same file. Refer to the online help for details.

Card Manager (RCA) Parameters

Table 25 - Card Manager (RCA) Product Parameters

Component	Parameter	Description and Default Value	Hot/Cold
ARM	UPDATE_CARD_CONTENT	Whether to update the Application Repository Manager Card Content. Default: False	C
ARM	CACHE_IDLE_LIFETIME	Application Repository database cache validity when operating in idle time (in seconds). Default: 20	C
ARM	CACHE_BURST_LIFETIME	Application Repository database cache validity when operating in burst mode (in seconds). Default: 600	C
ARM	CACHE_POLLING_PERIOD	Application Repository database cache polling period (in seconds). Set to zero to deactivate the cache. Default: 15	C
CATTP_MODULE	CONNECTION_INACTIVITY_PERIOD	Interval (in secs) at which a "keep alive" message is sent to the card. Default: 20	C
CATTP_MODULE	DEVICE_HANDLER_ADAPTOR	Indicates the name of the class that implements the DeviceCapabilitiesHandler interface. This class checks whether the subscriber's handset is BIP compatible. Refer to the <i>OTA Manager V5.1 Customization Guide</i> for further details. This value is checked at each service execution. If the value changes, the adapter is reloaded. Default: Empty	H
CATTP_MODULE	MAX_CATTP_WINDOW_SIZE	Maximum value for the Window Size indicated by the CATTP remote entity. If this parameter is not present, is equal to 0, or has a non-decimal value, no check is performed on the card window size. If this parameter has a valid value, the server considers that the card window size cannot be greater than this value. Note: A value of 1 results in a synchronous download.	H
CATTP_MODULE	MAX_CONNECTION_DURATION	Maximum time (in secs) for which communication between the card and the platform is maintained. This value must be greater than or equal to the CONNECTION_INACTIVITY_PERIOD parameter value. Default: 10*CONNECTION_INACTIVITY_PERIOD	C

Table 25 - Card Manager (RCA) Product Parameters (continued)

Component	Parameter	Description and Default Value	Hot/Cold
CATTP_MODULE	MAX_CONNECTION_INACTIVITY_TIME	Maximum period (in secs) of inactivity between the card and the platform. At the end of this period, the connection between the card and platform is stopped. This value must be greater than or equal to the CONNECTION_INACTIVITY_PERIOD. Default: 3*CONNECTION_INACTIVITY_PERIOD	C
CATTP_MODULE	MAX_CONNECTION_PULL_NBR	Maximum number of connections for MO requests. Default: 200	C
CATTP_MODULE	MAX_CONNECTION_PUSH_NBR	Maximum number of connections for Push commands. Default: 800	C
CATTP_MODULE	RETRANSMISSION_NB_MAX	Maximum number of retries for a protocol data unit (information packet). Default: 3	H
CATTP_MODULE	RETRANSMISSION_TIME_OUT	Wait time (in secs) before resending an unacknowledged protocol data unit (information packet). Default: 10	C
CATTP_MODULE	SMS_PUSH_ALPHA_ID	Message that displays on the mobile handset while waiting for the PDU to be executed. If no value is specified, a round robin on existing drivers is performed Default: null	H
CATTP_MODULE	SMS_PUSH_BEARER_DESC	Name of the network type used for messaging. For example, GPRS/UMTS. If no value is entered, the field is not included in the command. Default: null	H
CATTP_MODULE	SMS_PUSH_BUFFER_SIZE	Size of the information packet to be handled in a single session. If no value is entered, the field is not included in the command. Default: null	H
CATTP_MODULE	SMS_PUSH_CATTP_DEFAULT_DEST_PORT	Destination CAT-TP port used by the card for opening a communications channel. Default: 10	H
CATTP_MODULE	SMS_PUSH_CHANNEL_ID	The SMSC channel number to be used for Push commands. Default: null	H
CATTP_MODULE	SMS_PUSH_NETWORK_NAME	Network name. If no value is entered, the field is not included in the command. Default: null	H

Table 25 - Card Manager (RCA) Product Parameters (continued)

Component	Parameter	Description and Default Value	Hot/Cold
CATTP_MODULE	SMS_PUSH_OTA_IP_ADDRESS	IP address of the platform used for CAT_TP messaging. Format = xxx.xxx.xxx.xxx If no value is entered, the field is not included in the command. Default: null	H
CATTP_MODULE	SMS_PUSH_USER_LOGIN	User login name sent with the push command. If no value is entered, the field is not included in the command. Default: null	H
CATTP_MODULE	SMS_PUSH_USER_PWD	User password sent with the push command. If no value is entered, the field is not included in the command. Default: null	H
CATTP_MODULE	SMS_PUSH_UDP_PORT	Port on the platform used for CAT-TP messaging. This must match one of the UDP_SERVERS entries. If no value is entered, the field is not included in the command. Default: null	H
CATTP_MODULE	SMS_PUSH_WITH_POR	Whether or not a Proof Of Receipt (PoR) request is sent with the open connection command. Possible values: NONE, AS_SERVICE. AS_SERVICE uses same parameters as used for service requests. Default: null	H
CATTP_MODULE	SMS_PUSH_CLASS_CUSTOM	Name of the class that extends SmsPushHandler class. This class provides generatePush method which needs to be overwritten. A custom sms push data construction can be created. This value is checked at each service execution. Default: null Refer to the <i>OTA Manager V5.1 Customization Guide</i> for further details.	H
CATTP_MODULE	SUBSCRIBER_STATUS_HANDLER_ADAPTOR	Indicates the name of the class that implements the SubscriberStatusHandler interface. This class checks whether the subscriber's card is reachable. Refer to the <i>OTA Manager V5.1 Customization Guide</i> for further details. This value is checked at each service execution. Each time the value changes, the adapter is reloaded.	H

Table 25 - Card Manager (RCA) Product Parameters (continued)

Component	Parameter	Description and Default Value	Hot/Cold
CATTP_MODULE	UDP_SERVERS	List of UDP (User Datagram Protocol) servers. In the current release, only one server is supported. The IP address must match the SMS_PUSH_OTA_IP_ADDRESS product parameter value, if any. The port must match the SMS_PUSH_UDP_PORT product parameter value, if any. Format: (ipaddress:port) separated by blank characters. Default: null	C
CHANNELMONITORSMS	ACTIVE	Whether to activate the SMS channel monitor (Yes or No). Default: Yes	C
CHANNELMONITORWIR	ACTIVE	Whether to activate the WIR channel monitor (Yes or No). Default: Yes	H
CI	AC_TIMEOUT	Timeout, in milliseconds. The time a session must be inactive before it is automatically ended. Minimum value: 600000. Default: 600000	H
CI	QUEUES_SIZE	Maximum number of messages allowed in the queue. Valid values: 10-10000 Default: 100	H
CI	RETRY_DELAY	Delay (in ms) between each retry attempt. Valid values: 1000 to 10000. Default: 1000	H
CI	RETRY_NBR	Number of retries allowed when sending a message. Valid values: 1 to 3. Default: 3	H
GENERAL	DB_PSWD	Password used to access the Card Manager database. Default: <password_rca>	C
GENERAL	DB_URL	URL of the Card Manager database. Only THIN driver mode is supported. Default: jdbc:oracle:thin:@RCA	C
GENERAL	DB_USER	User connected to the Card Manager database. Default: <user_rca>	C
GENERAL	DB_VIEW_MESSAGES	Specifies whether or not to log Card Manager database transactions. Applies only to Message Builder module SQL logs. Default: False (do not log)	C

Table 25 - Card Manager (RCA) Product Parameters (continued)

Component	Parameter	Description and Default Value	Hot/Cold
GENERAL	DISABLE_CMM	If True, the Card Manager does not connect to the Campaign Manager and it is impossible to perform campaigns using the Card Manager. If False, or the parameter is missing, the Card Manager connection to the Campaign Manager is active and campaigns can be executed. Default: False for the default Card Manager instance, True for the Card Manager instance dedicated to XCT campaigns	C
GENERAL	JDBC_DRIVER	Oracle driver used for Java Database Connectivity. Default: oracle.jdbc.driver.OracleDriver	C
GENERAL	RCA_LOCATION	Directory in which Card Manager instances are installed. Default: <install_dir>/RCA	C
GENERAL	PLUGIN_LOCATION	The “plug-in” directory of the OTA Manager V5.1 platform. Default: <install_dir>/RCA/PlugIn/service	C
GENERAL	REPORTS_LOCATION	Directory in which the Card Manager writes all batchload reports. Default: <install_dir>/RCA/<instance>/name/logs	H
GENERAL	SHUTDOWN_TMEOUT	Time lapse (in minutes) before the Card Manager shuts down a module. The Card Manager uses this time for shutting down tasks and processing as many pending requests as possible. Default: 2	H
GENERAL	CMM_CACHE_LOCATION	Location of the temporary file that stores results of invocations launched by the Campaign Manager but that it cannot retrieve (for example, if the Campaign Manager is not reachable). Default: none	C
GENERAL	VALIDATE_XML	Specifies whether the server validates all XML provisioning files against the DTD cardframework.dtd. True to specify validation. Values: True, False Default: True for a new installation, false for migrations.	H
INVOCATION	PROTOCOL	Default protocol of an invocation. This protocol will be used when a “pushed” invocation does not have a protocol information defined. Values: SMS, WIR Default: SMS	C

Table 25 - Card Manager (RCA) Product Parameters (continued)

Component	Parameter	Description and Default Value	Hot/Cold
INVOCATION	UPDATE_ON_EXPIRED	<p>Specifies whether the synchronization counter on the card is updated if a service request invocation is in either the EXPIRED or SUPPRESSED state.</p> <p>Values: True, False</p> <p>Default: True</p>	H
INVOCATION	SUCCESS_ON_POR_EXPIRED	<p>Specifies whether an invocation is considered to be successful if a PoR was requested from the card and the specified timeout waiting for the PoR has expired.</p> <p>This causes a card content update if the service is configured accordingly.</p> <p>If the PoR is required on Error for some invocation, set SUCCESS_ON_POR_EXPIRED to true.</p> <p>The status of the invocation if a PoR is not received is made independently of the type of PoR required (PoR always, on Error).</p> <p>Values: True, False</p> <p>Default: True</p>	H
INVOCATION	GRACE_PERIOD_POR	<p>Amount of time (in seconds) that the platform waits for a PoR MO before assigning a final status to the invocation.</p> <p>This parameter is related to the SUCCESS_ON_POR_EXPIRED parameter.</p> <p>Values: integer number of seconds</p> <p>Default: -1</p> <p>If the value is set to -1, the platform waits for the PoR until the invocation expiration time (based on the invocation validity period).</p>	H
INVOCATION	VAL_PERIOD	<p>Default validity period (in ms) of an invocation. This value is used when a "pushed" invocation does not have a validity period defined.</p> <p>Values: 60 to "INFINITE".</p> <p>Default: INFINITE</p>	C
INVOCATION	SYNCH_COUNTER_CATTP_MANAGEMENT	<p>The method to use when the synchronization counter is updated for a CATTP invocation.</p> <p>Possible values:</p> <ul style="list-style-type: none"> ■ Normal. Increments the counter for each message with a received acknowledgement. ■ Optimist. Updates the counter when at least one message is sent to the card. <p>Default: Normal</p>	H

Table 25 - Card Manager (RCA) Product Parameters (continued)

Component	Parameter	Description and Default Value	Hot/Cold
INVOCATION	SYNCH_COUNTER_SMS_MANAGEMENT	<p>The method to use when the synchronization counter is updated for an SMS or WIR invocation.</p> <p>Possible values:</p> <ul style="list-style-type: none"> ■ Normal. Increments the counter for each message with a received acknowledgement. ■ Optimist. Updates the counter when at least one message is sent to the card. <p>Default: Normal</p>	H
INVOCATION	REPLAY_INVOCATION_DELAY_AT_RCA_RESTART	<p>Delay (in ms) between loading successive blocks of 50 incomplete invocations when the Card Manager (RCA) restarts.</p> <p>Warning: Too small a value can lead to system saturation.</p> <p>Values: 0 to 10000</p> <p>Default: 10000</p>	C
INVOCATION	REPLAY_INV_BLOCK_SIZE_AT_RCA_RESTART	<p>Size of the successive blocks of incomplete invocations loaded from the database when the Card Manager (RCA) restarts.</p> <p>Warning: Too large a value can lead to system saturation.</p> <p>Values: 50 to 1000 (invocations)</p> <p>Default: 50</p>	C
INVOCATION	REPLAY_INV_THROUGHPUT_AT_RCA_RESTART	<p>Provides a convenient method of specifying the authorized maximum flow rate of invocations following a restart of the Card Manager (RCA).</p> <p>If no value is set, the parameters REPLAY_INV_BLOCK_SIZE_AT_RCA_RESTART and REPLAY_INVOCATION_DELAY_AT_RCA_RESTART are used to calculate the resubmission rate.</p> <p>Values: 1 to 500</p> <p>Unit: Invocations per second</p>	H
INVOCATION_MANAGER	REGISTRY_USE	<p>Activate or deactivate access to the FM registry for this Card Manager. If False, no request is registered and therefore request monitoring, proof of execution, and Delivery Status messages are not available.</p> <p>Values: False, True</p> <p>Default: True</p>	C
INVOCATION_MANAGER	SEQUENCER_SIZE	<p>Stack size, in number of invocations, of the invocation manager sequencer for each invocation priority (NORMAL, URGENT, and EMERGENCY). The role of the sequencer is to manage priorities, to schedule requests, to order requests by expiration date and to manage request flow control.</p> <p>Default: 1000</p>	C

Table 25 - Card Manager (RCA) Product Parameters (continued)

Component	Parameter	Description and Default Value	Hot/Cold
INVOCATION_MANAGER	IM_CACHE_LOCATION	Location of the temporary file that stores pending invocation statuses managed when the FRWK database cannot be reached. Default: <i>install_dir/RCA</i>	C
JOB_PROCESSOR	JOB_QUEUE_SIZE	Maximum number of jobs allowed in the queue. Default: 800	C
JOB_PROCESSOR	MEMORY_SIZE	Maximum number of jobs allowed in memory. Default: 2000	C
JOB_PROCESSOR	NB_THREADS	Number of threads for processing the jobs in the Job queue. Default: 20	C
JOB_PROCESSOR	SWAP_SIZE	Maximum number of jobs allowed in the swap memory (SSA). Default: 100000	C
JOB_PROCESSOR	SWAP_TYPE	Type of swap memory. Values: DATABASE, MEMORY. Default: DATABASE	C
JOB_PROCESSOR	MIN_IN_MEMORY_TIME	Sets the minimum amount of time a job should stay in the primary area before considered as out of coverage (during the SMS sending phase) Default: 0	C
MESSAGE_BUILDER	DB_URL	URL of the Message Builder database. Only the THIN driver mode is supported. Default: <code>jdbc:oracle:thin:@RCA</code>	C
MESSAGE_BUILDER	DB_PSWD	Password used to connect to the Message Builder database. Default: <password_mb>	C
MESSAGE_BUILDER	DB_USER	User connected to the Message Builder database. Default: <user_mb>	C
MESSAGE_BUILDER	INSERT_COMMIT_SIZE	Size of the commit block when inserting security data in the Message Builder database. Default: 1000	C
MESSAGE_BUILDER	MAX_MESSAGES	Maximum number of messages that the formatting library is able to generate when executing a service. Do not modify this value. Default value: 100	C
MESSAGE_BUILDER	REQUEST_TIMEOUT	Maximum time allowed to perform an SQL request before considering that it has failed.	C
MO_ROUTING	MO_MEMORY_SIZE	Maximum number of MOs allowed in the SMSC mediator memory. Minimum value: 1000 Default: 1000	C

Table 25 - Card Manager (RCA) Product Parameters (continued)

Component	Parameter	Description and Default Value	Hot/Cold
MO_ROUTING	MO_PERSISTENCE	<p>Define whether or not a persistent swap memory is used.</p> <p>Values:</p> <ul style="list-style-type: none"> ■ ON - Swappable memory is persistent ■ OFF - Swappable memory is not persistent <p>Default : OFF</p>	C
MO_ROUTING	MO_QUEUE_MEMORY_SIZE	<p>Maximum number of complete MOs allowed in the MO processing memory queue. A complete MO means either a single MO or a reconcatenated MO.</p> <p>Minimum value: 1000</p> <p>Default: 1000</p>	C
MO_ROUTING	MO_QUEUE_SWAP_SIZE	<p>Maximum number of complete MOs allowed in the MO processing swap memory (SSA) queue.</p> <p>Minimum value: 1000</p> <p>Default: 100000</p>	C
MO_ROUTING	MO_SWAP_SIZE	<p>Maximum number of MOs allowed in the SMSC mediator swap memory (SSA).</p> <p>Minimum value: 1000</p> <p>Default: 100000</p>	C
MO_ROUTING	MO_UNFORMAT_SERVICE	<p>Name of the service to be used to unformat received MO messages.</p> <p>Default: Unformat</p>	C
MO_ROUTING	NB_THREADS	<p>Number of threads used to concatenate and unformat an MO message.</p> <p>Minimum value: 1</p> <p>Default: 10</p>	C
MO_ROUTING	MO_VALIDITY_PERIOD	<p>Amount of time (in milliseconds) that the platform waits for a concatenated message to be completed.</p> <p>Default: 86400</p>	C
PROTOCOLSMS	DCS_BINARY	<p>Default data coding scheme (in hexadecimal format) used with the SEND_BINARY command:</p> <ul style="list-style-type: none"> ■ F6 (8 byte - Class 2) ■ F4 (8 byte - Class 0) ■ F2 (7 byte - Class 2) ■ F0 (7 byte - Class 0) <p>Default: F6</p>	H
PROTOCOLSMS	DCS_ETSI_CARD	<p>Default data coding scheme (in hexadecimal format) used with the SEND_TEXT command in the case of an ETSI text for the card.</p> <p>Default: 00</p>	H
PROTOCOLSMS	DCS_ETSI_MOBILE	<p>Default data coding scheme (in hexadecimal format) used with the SEND_TEXT command in the case of an ETSI text for the mobile</p> <p>Default: 11</p>	H

Table 25 - Card Manager (RCA) Product Parameters (continued)

Component	Parameter	Description and Default Value	Hot/Cold
PROTOCOLSMS	DCS_UCS2_CARD	Default data coding scheme (in hexadecimal format) used with the SEND_TEXT command in the case of a UCS2 text for the card. Default: 08	H
PROTOCOLSMS	DCS_UCS2_MOBILE	Default data coding scheme (in hexadecimal format) used with the SEND_TEXT command in the case of a UCS2 text for the mobile. Default: 19	H
PROTOCOLSMS	DEFAULT_ENCODING	Format of destination data block. Valid values: UCS2 and ETSI. The following types of character substitutions are supported: <ul style="list-style-type: none">■ UNICODE to UCS2■ UNICODE to ETSI. Default: ETSI	H
PROTOCOLSMS	DRIVER_ACCEPT_DELAY	Time (in milliseconds) before the driver indicates that it is trying to post a new SMS to send. If the value is 0 (or parameter is missing), the previous product version's behaviour is retained. Default: 0	C
PROTOCOLSMS	MODE	Mode of the request. Valid values: <ul style="list-style-type: none">■ SMSC: A message must be received by the SMSC before another message is sent.■ MT (Mobile Terminating): A message must be received by the mobile before another message is sent.■ TRANS (Transaction): No Delivery Status message is sent from the SMSC. Default: TRANS	H
PROTOCOLSMS	MSG_LENGTH	Length of each message in the data block. Default: 140	H
PROTOCOLSMS	MSG_VAL_PERIOD	Validity period (in seconds) passed to the SMSC. This parameter is used only if the "current date + message validity period" is before the "invocation expiration date". Default: None. If no value is specified, the invocation validity period is used.	H
PROTOCOLSMS	ORIGINATING_ADDRESS	Refer to the TP-OA definition in the GSM 03.40 standard. Default: 0000	H
PROTOCOLSMS	PACKET_SENDING_RETRY	Time lapse (in seconds) before attempting to retry sending an invocation if the channel driver is disconnected, deactivated, or busy. Minimum value: 60 Default: 60	H

Table 25 - Card Manager (RCA) Product Parameters (continued)

Component	Parameter	Description and Default Value	Hot/Cold
PROTOCOLSMS	PID_BINARY	Identifier (in hexadecimal format) of the default protocol used with the SEND_BINARY command. Default: 7F	H
PROTOCOLSMS	PID_TEXT	Identifier (in hexadecimal format) of the default protocol used with the SEND_TEXT command. Default: 00	H
PROTOCOLSMS	TONNPI_INT_DEST	Type of Number/Numbering Plan Identifier used in international destination messages. Default: 91	H
PROTOCOLSMS	TONNPI_INT_ORIG	Type of Number/Numbering Plan Identifier used in international originating messages. Default: 91	H
PROTOCOLSMS	TONNPI_NAT_DEST	Type of Number/Numbering Plan Identifier used in national destination messages. Default: A1	H
PROTOCOLSMS	TONNPI_NAT_ORIG	Type of Number/Numbering Plan Identifier used in national originating messages. Default: A1	H
PROTOCOLWIR	DCS_BINARY	Data Coding Scheme for ESMS messages. Default: F6	H
PROTOCOLWIR	PID_BINARY	Protocol Identifier for ESMS messages. Default: 7F	H
PROTOCOLWIR	TONNPI_INT_ORIG	International TON/NPI for the originating address of an ESMS message. Default: 91	H
PROTOCOLWIR	TONNPI_NAT_ORIG	National TON/NPI for the originating address of an ESMS message. Default: A1	H
SIMCARD_VIEWER	CURSOR_TIMEOUT	Used to configure a database connection in read mode for a search operation.	C
SIMCARD_VIEWER	DB_CONNECTION_NBR	Used to configure a database connection in read mode for a search operation.	C
SIMCARD_VIEWER	MAX_OPENED_CURSOR	Used to configure a database connection in read mode for a search operation.	C

Table 25 - Card Manager (RCA) Product Parameters (continued)

Component	Parameter	Description and Default Value	Hot/Cold
X15	CONNECTION_TIME_OUT	Time (in seconds) after which an unused connection to the XCT database is closed. Default value: 900	C
X15	JOB_TIME_OUT	Time (in seconds) after which the Card Manager (RCA) rejects a batch of jobs received from XCT Default value: 60	C
X15	NB_THREAD	Number of connection threads to the XCT database. The value defined is separated for pre- and post-processing. For example: 10 represents 10 pre-processing and 10 post-processing threads. Recommended value: 10	C

XCT Parameters

Table 26 - XCT Product Parameters

Component	Parameter	Description	Hot/Cold?
DATABASE	REPOSITORY_DB_PASSWORD	Oracle password of the Campaign Manager database.	C
DATABASE	REPOSITORY_DB_URL	URL of the Campaign Manager database	C
DATABASE	REPOSITORY_DB_USERNAME	Oracle user name of the Campaign Manager database	C
DATABASE	X15_DB_URL	URL of the XCT database	C
DATABASE	X15_DB_USERNAME	Oracle user name of the XCT database	C
DATABASE	X15_DB_PASSWORD	Oracle password of the XCT database	C
PARAMETERS	RCA_ID	RCA CORBA identifier for pre- and post-processing. If the RCA_ID is "null", a round robin is performed on the OTA Manager instances before publishing the XCT plugin.	C
PARAMETERS	FLOW_CONTROL	Maximum number of cards being sent. Min.=SCHEDULER_FETCH_SIZE Max.=100,000,000 Default: 10,000	C
PARAMETERS	INVOCATION_CACHE_SIZE	Size of cache for invocations being sent. Used in the memory footprint calculation: see "Calculating the XCT Memory Footprint" on page 112. Min.=(DBFIFO_SIZE + DBWRITER_BATCH_SIZE + WINDOWS_SIZE) x NB_DBWRITER_THREAD Max.=6,000,000 Default: 50,000	C
PARAMETERS	SMS_CACHE_SIZE	Size of SMS cache. Used in the memory footprint calculation: see "Calculating the XCT Memory Footprint" on page 112. Valid values: 1 - 6,000,000 Default: 150,000	C
PARAMETERS	SENDING_PAUSE_MODE_FINISH_INVOCATIONS	Sending pause mode: Whether to keep sending SMSs to a particular card when the Delivery Status is received. Valid values: True/False Default: False	C
CHANNEL	WINDOW_SIZE	Maximum number of SMSs sent before receiving an acknowledgment from the SMSC. Valid values: 1-1000 Default: 10	C
CHANNEL	SMSC_IP_ADDRESS	Address of SMSC.	C
CHANNEL	SMSC_IP_PORT	Port number of SMSC.	C
CHANNEL	SMSC_PASSWORD	Password used to connect to the SMSC.	C
CHANNEL	SMSC_TYPE	Type of SMSC. Valid values = CMG, SMPP, Nokia	C

Table 26 - XCT Product Parameters (continued)

Component	Parameter	Description	Hot/Cold?
CHANNEL	SMSC_USER	Name of users connected to the SMSC.	C
CHANNEL	SS7_MODE	Whether to use SS7 mode. If False, all alerts are ignored. Valid values: True or False. Default: False	C
CHANNEL	TP_OA	Originating address (TP-OA) of the XCT platform.	C
CHANNEL	TP_OA_TON_NPI	TON/NPI of the platform's Originating Address. Default: 91	C
CHANNEL	TP_DA_TON_NPI	TON/NPI of card (Destination Address). Default: 91	C
CHANNEL	INQUIRY_ACTIVATED	Whether queries are sent to the SMSC when the validity period and grace period expire. Valid values: True or False (False for CMG). Default: True	C
CHANNEL	RETRY_NUMBER	Number of retries allowed when connecting to the SMSC to send MTs. Min.=0 Max.=10 Default: 3	C
CHANNEL	RETRY_TIMEOUT	Time (in secs) after which the driver resends the MT. Min.=10; if 0 then no retry is performed Max.=3600 Default: 10	C
CHANNEL	LOGIN_TIMEOUT	Timeout (in secs) after which the driver stops attempting to connect to the SMSC. Default: 10	C
CHANNEL	LOGIN_RETRY_PERIOD	Time (in secs) before the driver attempts to reconnect to the SMSC. Min.=10 Max.=3600 Default: 10	C
CHANNEL	ALIVE_PERIOD	Wait time (in secs) before an "alive" is sent. Min.=10; if 0, no "alive" is sent Max.=3600 Default: 0	C
CHANNEL	LOCAL_PORT	Local port of the platform. Default: 0	C
CHANNEL	STACK_SIZE	Internal stack size of SMS driver. Min.=1 Max.=10000 Default: 1000	C

Configuring SMSC Channel Drivers

This appendix describes how to configure SMSC channel drivers.

Note: You can manage SMSC channel drivers only when the **Platform Management** option has been specified in your user profile.

You can configure channel drivers by:

- 1 Modifying the Initialization String (“**InitString**”).
- 2 Configuring the channel driver’s initialization file (`drivers.ini`). See “Configuring Channel Drivers” on page 189.
- 3 Setting the channel driver velocity. See “Activating and Deactivating the Audit Trail” on page 32.

This appendix also lists the versions of SMSCs and SMSC protocols supported by OTA Manager V5.1.

Format of the Initialization String (**InitString**)

The channel driver sends an initialization string to the SMSC whenever the connection is made or reset. This string is different for each supported SMSC type (Nokia, CMG, Logica/SMPP), and is configurable.

The syntax of the **InitString** of the channel driver depends on the implementation that the user chooses.

For each channel driver, the format of the **InitString** is as follows:

`IPADDRESS:IPPORT:RETPNUMBER:RETTO:LOGNAME:LOGPASS:LOGRETPERIOD:WINDOW_SIZE:ALIVEPERIOD:LOGINTIMEOUT:STACKSIZE:LOCALPORT:IPMODE`

An example is shown below:

`polux:8101:3:1000:gemplus:toto:10000:1:60000:10000:2000:0:IP`

You can display the current value of the **InitString** initialization string on the Driver Management window.

The following table describes the parameters used by this string. All parameters are mandatory. For other optional parameters for each driver, see “Table ” on page 190.

Note: An asterisk (*) indicates that the parameter is mandatory.

Table 27 - Format of the InitString

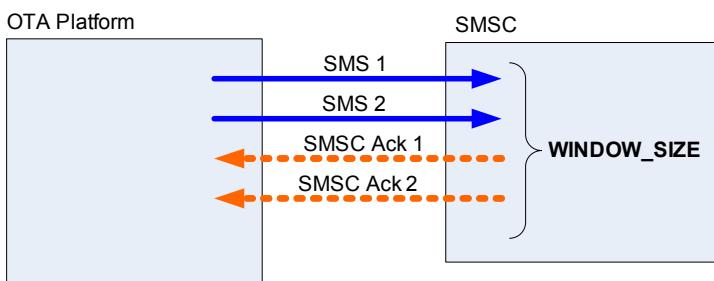
Parameter Name	Description
IPADDRESS*	IP address of the SMSC.
IPPORT*	IP port number of the SMSC.
RETRNUMBER	Number of attempts to retry delivery of an SMS when the SMSC does not respond (SMSC Ack). For classic campaigns, the recommended value is 0. For XCT campaigns, set this value to 3.
RETTO*	Time lapse (in ms) between two attempts to send a message when receiving no response from the SMSC. If no SMSC-ACK has been received when this period of time has elapsed, X attempts are made to send the message (see the RETNUMBER parameter) before deleting it from the queue. If the value is set to less than 1000, it is set to 1000.
LOGNAME*	Logon name used to connect to the SMSC. In the case of a CMG SMSC, 'NULL' in the logon name means that the driver should not use the UCP 60 message when logging on.
LOGPASS*	Logon password used to connect to the SMSC.
LOGRETPERIOD*	Amount of time (in ms) the driver waits when an error occurs, before trying to reconnect using the RETNUMBER value for the number of retry attempts.
WINDOW_SIZE*	The WINDOW_SIZE parameter specifies the maximum number of SMS messages that OTA Manager can send without waiting for the corresponding acknowledgment messages (SMSC Ack) to be returned by the SMSC. The following values are possible: 0 : No Waiting 1 : stop and wait mode n : send n messages without SMSC Ack before waiting. In the following example, the WINDOW_SIZE is 2:
	
<p>The initial value of the WINDOW_SIZE parameter is set by Gemalto during the integration phase. Its value depends on the SMSC type:</p> <ul style="list-style-type: none"> ■ On CMG, it must be set equal to the Windowing value of the SMSC. You should therefore ask your SMSC administrator for the value to use. ■ On Nokia, it is recommended to set it to 1. ■ On SMPP, a value of 30 can be used. If you detect problems, set it back to 1. <p>If the value is unknown (for example, no value was specified during integration), set the WINDOW_SIZE value to 1.</p> <p>Warning: It is NOT recommended to modify this parameter's value for individual campaigns. Only consider modifying the value after consultation with Gemalto support personnel.</p>	
ALIVEPERIOD*	Amount of time (in ms) between two live queries. If the ALIVEPERIOD value is 0, a live process is not started.

Table 27 - Format of the InitString (continued) (continued)

Parameter Name	Description
LOGINTIMEOUT*	Amount of time (in ms) that the driver waits for a logon response before the response is set to "negative".
STACKSIZE*	Maximum number of messages that can be stored in the driver before the driver is set to "busy". When the number of stored messages falls to 90% of STACKSIZE, the driver is set to "not busy". Refer to the Excel calculation file.
LOCALPORT*	Indicates the local port to be used when using an IP Protocol. Value 0 means that a dynamic local port should be used. It is also possible to indicate a local address by adding @ip. For example: <ul style="list-style-type: none">■ 12345 indicates a specific local port only■ 12345@10.10.100.100 indicates a specific local port and IP address■ @10.10.100.100 indicates a specific local address only
IPMODE*	Indicates whether the driver is used on an IP or on an X25 protocol. Values are 'IP' for IP protocol and the 'X25 card name' for an X25 Protocol.

Configuring Channel Drivers

The configuration file for the Card Manager drivers is \$PLATFORM_HOME/RCA/PlugIn/drivers/drivers.ini, where \$PLATFORM_HOME is the root installation directory of the platform.

The configuration file for XCT drivers is \$PLATFORM_HOME/XCT/PlugIn/drivers/drivers.ini. The XCT driver is referenced as [DRIVER1] in this file.

A third configuration file, drivers.ini.bmd, is available in the directory \$PLATFORM_HOME/XCT/PlugIn/drivers/driver.ini.bmd. This is a pre-configured driver configuration file for use with a BMD SMS router.

For the Nokia, CMG, and Logica/SMPP drivers, values for certain optional parameters may also be defined. These are not defined in the **initString**, but inserted in the appropriate configuration file.

The parameter values are read from this file for a particular channel driver when that channel driver is activated. As this implies, values may be changed for subsequent messages by deactivating the driver, editing this file, and then reactivating the driver.

Values are defined in the drivers.ini file using the following format. For example:

```
[driver2]
tariff_class=1
[driver4]
oton=7
onpi=3
```

The number of the driver (for example, driver2) is the Channel Identifier number as it appears in the management interface.

The optional parameters are listed in the following tables. If the parameter is not defined

in the drivers.ini file, the default value is used.

Table 28 - Optional Parameters Common for all Supported Drivers

Parameter	Default value
MO_REFUSED_POLICY This parameter indicates the error code to send to the SMSC if an MO handling error occurs in OTA Manager V5.1.	No Error
MO_TEMPORARY_REFUSED_POLICY This parameter indicates the error code to send to the SMSC if a temporary MO handling error occurs in OTA Manager V5.1.	No Error
HEARTBEAT_BACKWARD_COMPATIBILITY If set to "true", this parameter indicates that the driver is compatible with the behavior of previous versions with regard to the alive_period.	False
EXPIRED_CODE_LIST This is a space-separated list of SIM_ACK error codes that have been handled by OTA Manager V5.1 as expired error codes. For all codes NOT included in this list, no retry is performed.	Empty
FLOW_CONTROL_SLEEP This is the time (in ms) to sleep between sending two messages. Possible values: 20 to 200. See "Flow Control between OTA Manager and the SMSC" on page 197 for more information on this parameter.	0
FLOW_CONTROL_SLEEP_FREQUENCE The number of SMS messages after which to introduce the FLOW_CONTROL_SLEEP delay. See "Flow Control between OTA Manager and the SMSC" on page 197 for more information on this parameter.	0
SMS_VP_TIME_SHIFT This optional parameter corrects the time difference between the platform and SMSCs. This may be necessary when the validity period is short. Applies to all channel driver types that allow time shift management. The value is in seconds. For example: sms_vp_time_shift = -30 means remove 30 seconds from the validity period sms_vp_time_shift = 3600 means add 3600 seconds to the validity period.	-
POR_IN_DELIVER_REPORT If set to true, this optional parameter activates the reception of PoRs in delivery reports in the same command as a SIMACK message. This parameter is available for SMPP drivers and if the environment (for example, SMSC, Network) allows it. Note: This parameter cannot be set to true if IGNORE_OPTIONAL_PARAMETERS parameter is set to true.	False

Table 28 - Optional Parameters Common for all Supported Drivers (continued)

Parameter	Default value
ORIGINATING_ADDRESS This optional parameter indicates the value of Originating Address which will be sent in the message	Null
ADDRESS_RANGE This parameter is used in the bind_receiver and bind_transceiver command to specify a set of addresses serviced by the platform. A single address may also be specified.	
UNBIND_BEFORE_BIND Indicates that the channel driver must send a logout (unbind) to the SMSC before a login (rebind) whenever the connection has been lost. Possible values: true / false. Note: This parameter is only available for SMPP and Nokia channel drivers.	True

Table 29 - Optional Parameters for the Nokia Channel Driver

Parameter	Default value
TARIFF_CLASS	Null
SERVICE_DESCRIPTION	Null

Table 30 - Optional Parameters for the CMG Channel Driver

Parameter	Default value
OTON Defines the Originator Type of Number in the Login command. Possible values: <ul style="list-style-type: none">■ 1: International number (starts with the country code)■ 2: National number■ 6: Abbreviated number (short number alias). Default value if omitted.	6
ONPI Defines the Originator Numbering Plan Identifier in the Login command. Possible values: <ul style="list-style-type: none">■ 1: E.164 address■ 3: X121 address■ 5: Private (TCP/IP address/abbreviated number address). Default value if omitted.	5
VERS Defines Version number in the Login command Possible value: 0100	0100
OPID Defines the Originator Protocol Identifier in the Login command. Possible values: <ul style="list-style-type: none">■ 00: Mobile station■ 39: PC application■ Otherwise, the parameter value is Null.	Null
SMTPID	0539
ALIVE_MODE	ucp51 (ucp31, ucp51)
SINGLE_SHOT If set to "true", the driver uses the single shot indicator for all messages sent.	False

Table 30 - Optional Parameters for the CMG Channel Driver (continued)

Parameter	Default value
VERSION Indicates the SMSC version. It allows the driver to optimize usage of CMG features. Possible values are 3.1.0, 3.1.1, 3.1.2, 3.1.4, 3.5, and 4.0	4.0
BILLING_IDENTIFIER Note: This identifier is used for all messages sent.	No value
OTOA Defines the Originator Type Of Address in the Send Message command. Possible values: <ul style="list-style-type: none">■ If the value is 1139, the Address Code Originator (OAdC) is set to NPI telephone and TON international.■ If the value is 5039, the Address Code Originator (OAdC) contains an alphanumeric address.■ Otherwise, the parameter value is Null for a numeric address in the Address Code Originator (OAdC).	Null
NOTIFICATION_TYPE Whether to send notification when SMS could not be delivered but will try again later. Possible values are: 0=default, 1=delivery notification (DN) only, 2=non-delivery notification (ND) only, 3=DN and ND, 4=buffered message notification (BN), 5=BN and DN, 6=BN and ND, 7=DN, ND, and BN	0

Table 31 - Optional Parameters for the Logica/SMPP Channel Driver

Parameter	Default Value
SYSTEM_TYPE This optional parameter identifies the type of the system requesting to bind with the SMSC as a transmitter ("OTA" for example). Note: Specification of this parameter is optional; some SMSCs may not require this detail. In this case, the value is set to Null.	Null
SERVICE_TYPE This optional parameter indicates the SMS application service associated with the message to send. This parameter can also implicitly activate the "replace if present" functionality when the value used is the same as the one configured in the SMSC allowing the replace function. Note: The "Replace if present" functionality can be also activated by using the parameter REPLACE_MESSAGE_FLAG (see "REPLACE_MESSAGE_FLAG" on page 194).	Null
INTERFACE_VERSION This optional parameter indicates to the SMSC the version of the SMPP protocol used by the platform. This parameter is set in a BIND PDU sent to the SMSC.	34
ADDR_TON This optional parameter indicates to the SMSC the Type of Number (TON) address used by the Platform. This parameter is set in a BIND PDU sent to the SMSC.	0

Table 31 - Optional Parameters for the Logica/SMPP Channel Driver (continued)

Parameter	Default Value
ADDR_NPI This optional parameter indicates to the SMSC the Numbering Plan Indicator (NPI) address used by the platform. This parameter is set in a BIND PDU sent to the SMSC.	0
ADDRESS_RANGE	Null
TIMEZONE Format "mm +" or "mm -" Time difference in quarter hours between local time and UTC time + indicates local time is in quarter hours advanced in relation to UTC time - indicates local time is in quarter hours retarded in relation to UTC time	00+
VERSION Indicates the SMSC version. It allows the driver to optimize usage of SMPP features. Possible values: 3.3, 3.4, UNKNOWN	UNKNOWN
IGNORE_OPTIONAL_PARAMETERS If set to true, the driver does not use the optional parameters in deliver_sm (case of SIMACK) to get the message ID (0x001E), state (0x0427) and error (0x0423) fields. These fields are parsed from short_message or from message_payload parameter (if short_message size defined in sm_length parameter = 0). In this case the parameters are defined by 'id', 'stat:' and 'err:'. Possible values: true / false.	False
CDMA_SMPP_COMPLIANCE Indicates the driver complies with CDMA SMPP implementations (with a non-integer SMSC reference). If set to true, the SMSC reference is managed as string. Possible values: true / false	False
ADDR_LIST_WITHOUT_VP Indicates a list of originating addresses for which all messages coming from these addresses will have their validity period set to "Null". Each list item must be separated by a space character.	Empty

Table 31 - Optional Parameters for the Logica/SMPP Channel Driver (continued)

Parameter	Default Value
MESSAGE_MODE Indicates the message mode in 'esm_class' field of SUBMIT_SM or DATA_SM commands. Possible values: 0=Default SMSC mode, 1=Datagram mode, 2=Transaction mode, 3=Store and Forward mode. Note: <ul style="list-style-type: none">- MESSAGE_MODE parameter is available only in SMPP 3.4 version.- SUBMIT_SM command is incompatible with Transaction mode, this is why if MESSAGE_MODE=2, DATA_SM command is used for send message. (For more details see SMPP 3.4 specification version)	0
REPLACE_MESSAGE_FLAG Manages the possibility to replace message which has been previously submitted but has not yet been delivered to the destination. This parameter updates 'replace_if_present_flag' field of SUBMIT_SM command. Possible values: 0=Don't replace, 1=replace Note: This parameter is only available when the message is sent in MT mode.	0
CONVERT_SMSREF_IN_USER_DATA If set to true, the SMSC Reference parsed in user data field (short_message or message_payload) of DELIVER_SM command is converted. Possible values: true / false. Note: This parameter is taken into account when CDMA_SMPP_COMPLIANCE is set to true	False

Configuring “handle_temporary_status=true” in drivers.ini

In order to force the XCT module to interpret Delivery Status (DS) messages returned by the SMSC (to differentiate between temporary and permanent error codes), you must configure the following flag in the `drivers.ini` file for XCT:

```
[PRODUCT <XCT_product_name>]
[DRIVER1]
handle_temporary_status=true
```

Note: XCT only retries SMS messages for which a temporary error code has been returned by the SMSC.

Configuring EXPIRED_CODE_LIST in drivers.ini

The network error code may or may not be prefixed by a “0”, depending on the SMSC implementation. Since the driver compares network error codes as strings and not as integers, both values (with and without the “0” prefix) should be specified in the `drivers.ini` file’s “expired code list”.

Configuring RETRY_SMS_ON_NACK_RESPONSE in drivers.ini

This parameter allows you to activate a retry mechanism to resend messages upon receiving a negative response message.

When the platform receives a negative response (on SMSC-ACK) with a specific error code for the sent message, the SMS is submitted again to the destination without taking into account the number of retries allowed (defined by the parameter RETNUMBER—see “Table 27” on page 188).

This mechanism has been primarily developed for OTACS when configured in standalone mode. When OTACS loses the connection with the SMSC for messages sent by the platform, OTACS acknowledges with a NACK response containing a specific error code. The messages are not lost and are resubmitted until the connection with the SMSC is restored.

Available Protocols

This parameter is available for the SMPP and CMG protocols.

Available Error Codes in the NACK Response

When using the SMPP protocol, if the RETRY_SMS_ON_NACK_RESPONSE parameter is set to “true” and the error code 20 is received in the NACK response, the SMS is resubmitted to the destination.

When using the CMG protocol, if the RETRY_SMS_ON_NACK_RESPONSE parameter is set to “true” and the error code 4 is received in the NACK response, the SMS is resubmitted to the destination.

Available Parameter Values

RETRY_SMS_ON_NACK_RESPONSE = true|false

Default Value

The default value of the RETRY_SMS_ON_NACK_RESPONSE parameter is “false” for both protocols.

Configuring MSISDN_TX_CONVERT_FORMAT and MSISDN_RX_CONVERT_FORMAT in drivers.ini

Two parameters, MSISDN_TX_CONVERT_FORMAT and MSISDN_RX_CONVERT_FORMAT, are used to manage MSISDN conversion in channel drivers.

These parameters are defined in the `drivers.ini` file and are used for SMPP, CMG and NOKIA drivers.

The MSISDN_TX_CONVERT_FORMAT common channel driver configuration parameter defines rules for converting MSISDN destination addresses in messages being sent by the platform.

The MSISDN_RX_CONVERT_FORMAT common channel driver configuration parameter defines rules for converting MSISDN source addresses received in SIM Ack or MO command messages.

Format:

MSISDN_TX_CONVERT_FORMAT	=	[AT <i>index</i> = XXX] or [AT <i>index</i> != XXX]	[SUP= <i>index1</i> <i>index2</i> ...]	[ADD= <i>index1:value1</i> <i>index2:value2</i> ...]
---------------------------------	---	---	--	--

(See Note 1)

(See Note 2)

(See Note 3)

Notes:

- 1 **AT tag:** Defines the condition for converting the MSISDN:
 - *index*: indicates the position within the MSISDN
 - Equal (“=”) and Not Equal (“!=”) operators are available.
 - XXX: is the value to be compared, it can be digits or a null identifier.
- 2 **SUP tag:** Defines indexes of digits to remove in the MSISDN if the condition is satisfied. It is possible to suppress several indexes by using the separator “|”.

3 ADD tag: Defines the digits to add at corresponding indexes of MSISDN if the condition is satisfied. It is possible to add several indexes by using the separator “|”.

Remarks:

- The index begins at zero (0).
- The tags are processed in the following order:
 - a) Condition tag **[AT]**
 - b) Suppression tag **[SUP]**
 - c) Add tag **[ADD]**
- The **[AT]** condition is optional. If it exists, either the suppression tag **[SUP]**, the add tag **[ADD]**, or both, must be defined. Otherwise, if the condition does not exist, the conversion is applied regardless of the MSISDN value.

Note: In the following examples, if the condition is not satisfied, the MSISDN is not converted.

Example 1

MSISDN_TX_CONVERT_FORMAT= [AT 0 != 0049] [SUP=0] [ADD=0:0 | 1:0 | 2:4 | 3:9] (For sending message)

MSISDN_RX_CONVERT_FORMAT= [AT 0 != 0049] [SUP=0] [ADD=0:0 | 1:0 | 2:4 | 3:9] (For receiving message)

The MSISDN is converted if it does not start with 0049, in which case the first digit is removed and the digits “0049” are prefixed to the MSISDN:

MSISDN = 123456789

(Condition satisfied) **123456789 => (Apply SUP tag) 23456789 => (Apply ADD Tag) 004923456789**

Example 2

MSISDN_TX_CONVERT_FORMAT = [AT 9=null] [ADD=0:8] (For sending message)

MSISDN_RX_CONVERT_FORMAT = [AT 9=null] [ADD=0:8] (For receiving message)

The MSISDN is converted if the 9th index does not exist; in this case, the digit “8” is prefixed to the MSISDN:

MSISDN = 123456789

(Condition satisfied) **123456789 => (Apply ADD tag) 8123456789**

Example 3

MSISDN_TX_CONVERT_FORMAT = [AT 0 = 0033] [SUP = 0 | 1 | 2 | 3] (For sending message)

MSISDN_RX_CONVERT_FORMAT = [AT 0 = 0033] [SUP = 0 | 1 | 2 | 3] (For receiving message)

The MSISDN is converted if it starts with “0033”; in this case, the four digits are removed from the beginning of the MSISDN:

MSISDN = 0033123456789

(Condition satisfied) **0033123456789 => (Apply SUP tag) 123456789**

Example 4

MSISDN_TX_CONVERT_FORMAT = [SUP = 0 | 1] [ADD=0:1] (For sending message)

MSISDN_RX_CONVERT_FORMAT = [SUP = 0 | 1] [ADD=0:1] (For receiving message)

Whatever the MSISDN destination address, the two first digits are removed and the digit “1” is prefixed to the MSISDN:

MSISDN = 00123456789

00123456789 => (Apply SUP Tag) 123456789 => (Apply ADD tag) 1123456789

Flow Control between OTA Manager and the SMSC

Normally, flow control is dynamically controlled by the SMSC, by adding a delay to the SUBMIT-RESPONSE packets. For example, if the “window size” is set to 10, OTA Manager V5.1 immediately sends the first 10 SMS messages (SUBMIT packets from OTA Manager V5.1 to SMSC), then waits for the reception of SUBMIT-RESPONSE packets (from the SMSC to OTA Manager V5.1) before sending the next SMS message. Any delay dynamically introduced by the SMSC will therefore lead to a reduction in throughput on the OTA Manager V5.1 side.

For this reason, it is possible to statically limit the throughput of SMS messages leaving OTA Manager V5.1 using the following parameters in the `drivers.ini` file:

- The `FLOW_CONTROL_SLEEP` parameter in the `drivers.ini` file introduces a delay by OTA Manager V5.1 after the sending of each SMS. The value can be configured between 20 milliseconds and 200 milliseconds. If not set (flow control sleep is 0 milliseconds), no throughput limit is set.

For example:

```
[DriverX]flow_control_sleep= 50
```

(sets flow control sleep of 50 milliseconds, resulting in a throughput of around 20 SMS/second).

- `FLOW_CONTROL_SLEEP_FREQUENCE`: if this parameter is specified, the `FLOW_CONTROL_SLEEP` parameter value is not introduced after the sending of each SMS message, but only after the number of SMS messages specified as the `FLOW_CONTROL_SLEEP_FREQUENCE` parameter value. The delay introduced is the value specified in the `FLOW_CONTROL_SLEEP` parameter.

Note: The “flow control sleep” mechanism should be considered as a workaround solution: Gemalto cannot guarantee the correlation between the “flow control sleep” values and the corresponding throughput achieved: this must be tuned in the operator’s environment. The preferred solution for flow control between OTA Manager and the SMSC remains the use of a “window size” and a dynamic delay on SUBMIT-RESPONSE packets introduced by the SMSC.

Pre-configured drivers.ini for Integration with BMD SMS Router

OTA Manager V5.1 has been tested with the BMD SMS router. A pre-configured `drivers.ini.bmd` file is available in the following directory:

```
$PLATFORM_HOME/XCT/PlugIn/drivers/
```

If your XCT is integrated with a BMD SMS router, simply rename `drivers.ini.bmd` to `drivers.ini`.

Determining Channel Driver Parameters (`drivers.ini`)

Supported SMSC Versions and Protocols

OTA Manager V5.1 supports the following SMSC versions and protocols.

Nokia

CIMD2 (Nokia SMSC Protocol) release SC5 A

CIMD2 (Nokia SMSC Protocol) release SC5 B

SMPP

SMPP v3.4 is provided.

Note: The minimum version supported by OTA Manager V5.1 is SMPP v3.4. OTA Manager V5.1 is not natively compatible with the SMPP 3.3 protocol.

There is a workaround that has been used with previous versions of OTA Manager V5.1: it consists of instantiating two SMPP 3.4 drivers: one for the Transmitter connection (TX), and one for the Receiver connection (RX).

Note that this workaround can still be used in the context of the Card Manager (RCA), but it cannot be used with XCT, because XCT has only a single SMSC channel driver. It may be possible to use OTACS as a protocol translator between XCT (SMPP 3.4, Transceiver mode: TRX) and SMPP 3.3 SMSC in Transmitter+Receiver mode (TX+RX), but this configuration must be tested.

CMG

UCP (LCMG SMSC Protocol) 3.1.0

UCP (LCMG SMSC Protocol) 3.1.1

UCP (LCMG SMSC Protocol) 3.1.4

UCP (LCMG SMSC Protocol) 3.5

UCP (LCMG SMSC Protocol) 4.0

Log File, Audit Trail, and Billing Ticket File Formats

Logging

The Logging facility provides a history of OTA Manager V5.1 platform activity. Logging can also be used for debugging in both development and production environments.

Format of the Log File

Log file records can be generated in either CSV (Comma Separated Value) or XML format.

CSV File Format

Each log file is made up of a single header line followed by zero or more lines containing log entries. Lines are separated by a new line character ("\n"). Fields within each record are separated by a comma (",").

The header line has the following format:

LOGGING_FILE,*creationDate*

The format of each line is as follows:

productName,date,level,originatingID,address,methodName,processID,productType,errorCode,freeText

The log file parameters are:

Table 32 - Log File Fields

Log Parameter	Description
<i>productName</i>	Product name, for example "RCA1"
<i>date</i>	Date and time of event in the format: YYYY/MM/DD HH:MM:SS:mmm
<i>level</i>	Trace level of the event
<i>textualTraceLevel</i>	Text equivalent of the trace level (FATAL, ERROR, WARN, INFO, or DEBUG).
<i>DomainName</i>	Name of the logging domain that generated the event.
<i>methodName /Class</i>	Name of the method in which the event occurred
<i>thread ID</i>	Identifier of the thread where the logged event occurred (optional)

Table 32 - Log File Fields (continued)

Log Parameter	Description
<i>productType</i>	Product in which the event occurred (optional)
<i>errorCode</i>	Error code associated with the event (optional)
<i>freeText</i>	Text description associated with the event (optional).

Example

The following is an example of a log file in CSV format:

```
RCA1,2005/02/07 15:00:00,8,INFO,DriverSmsc6,Trace.info,thread(2387135)=
DriverSmsc6,RCA1,0,Trying to connect nbr 5 at e000277:8003
RCA1,2005/02/07 15:00:00,8,INFO,ChannelMonitorSms,Trace.info,thread(7182897)=
JobProcessor/PT#11,RCA1,0,Sending message: :35101EC31D422#MSG0
RCA1,2005/02/07 15:00:00,8,INFO,ChannelMonitorSms,Trace.info,thread(7182897)=
JobProcessor/PT#11,RCA1,0,Sending message: :35101EC691BBF#MSG0
RCA1,2005/02/07 15:00:00,8,INFO,ChannelMonitorSms,Trace.info,thread(2387135)=
DriverSmsc6,RCA1,0,receiveSmscAck() called
RCA1,2005/02/07 15:00:00,8,INFO,ChannelMonitorSms,Trace.info,thread(2387135)=
DriverSmsc6,RCA1,0,Posting SmscAckEvent (35101EC691BBF, null, 0,
ChannelDriverSMSEception: [errorCode:709] ErrorDeconnection)
RCA1,2005/02/07 15:00:00,8,INFO,ChannelMonitorSms,Trace.info,thread(2387135)=
DriverSmsc6,RCA1,0,Posting SmscRetryEvent (35101EC691BBF, 0)
RCA1,2005/02/07 15:00:00,8,INFO,ChannelMonitorSms,Trace.info,thread(3233200)=
JobProcessor/PT#18,RCA1,0,send() called
RCA1,2005/02/07 15:00:00,8,INFO,ChannelMonitorSms,Trace.info,thread(6334563)=
JobProcessor/PT#19,RCA1,0,send() called
RCA1,2005/02/07 15:00:00,8,INFO,ChannelMonitorSms,Trace.info,thread(3233200)=
JobProcessor/PT#18,RCA1,0,Sending message: :35101ED1C49AF#MSG0
RCA1,2005/02/07 15:00:00,8,INFO,ChannelMonitorSms,Trace.info,thread(2387135)=
DriverSmsc6,RCA1,0,receiveSmscAck() called
RCA1,2005/02/07 15:00:00,8,INFO,DriverSmsc6,Trace.info,thread(2387135)=
DriverSmsc6,RCA1,0,Trying to connect nbr 4 at e000277:8003
RCA1,2005/02/07 15:00:00,1,ERROR,DriverSmsc6,Trace.error,thread(2387135)=
DriverSmsc6,RCA1,0, EX=[DriverConnectionException: [errorCode:718] e000277:8003]
RCA1,2005/02/07 15:00:00,1,ERROR,DriverSmsc6,Trace.error,thread(2387135)=
DriverSmsc6,RCA1,0, EX=[DriverConnectionException: [errorCode:718] e000277:8003]
```

XML File Format DTD

The following DTD describes the XML format of log files:

```
<!ELEMENT file ( header+, LINE* ) >
<!ELEMENT header ( title, date ) >
<!ELEMENT title ( #PCDATA ) >
<!ELEMENT date ( #PCDATA ) >
<!ELEMENT LINE ( freeText ) >
<!ATTLIST LINE ProductName #REQUIRED >
<!ATTLIST LINE bsDate CDATA #REQUIRED >
<!ATTLIST LINE level NMTOKEN #REQUIRED >
<!ATTLIST LINE TextualTraceLevel NMTOKEN #REQUIRED >
<!ATTLIST LINE DomainName NMTOKEN #REQUIRED >
<!ATTLIST LINE methodName NMTOKEN #REQUIRED >
<!ATTLIST LINE threadId NMTOKEN #REQUIRED >
<!ATTLIST LINE productType NMTOKEN #REQUIRED >
<!ATTLIST LINE eventId NMTOKEN #REQUIRED >
```

```
<!ELEMENT freeText ( #PCDATA ) >
```

XCT Log Files

The following trace domains are possible:

Table 33 - XCT Log Files

Trace domain	Description	Comments
RootTrace	Traces the life cycle of XCT product (start, shutdown and kill)	This domain is not verbose
CommunicationInterface	Not used	
X15	Trace management of time slots for different phases (pre-processing, sending and post-processing)	This domain is not verbose
performance (see note following table)	Traces the following information every 10 seconds: <ul style="list-style-type: none"> ■ Current invocation in SENDING state ■ Average throughput of Submit Response reception ■ Average throughput of Delivery Status reception ■ Hit ratio on invocation cache and sms cache ■ Minimum, maximum and average time to terminate an invocation 	This domain is fairly verbose. This displays only database contentions
postprocessing	Trace scheduler of XCT post-processing module	This domain is not verbose
preprocessing	Trace scheduler of XCT pre-processing module	This domain is not verbose
sending	Trace state chart of the XCT sending module	This domain is very verbose
cachemanager	Trace invocation or SMS cache manager. If neither invocation nor SMS are not present in the cache, information is retrieved from the database.	This domain is very verbose
scheduler	Trace scheduler of XCT sending module which gets invocations and messages from the database.	This domain is not verbose if FLOW_CONTROL is not reached
smsgateway	Traces actions between XCT and the channel driver: receipt of Submit Response, Delivery Status and MO.	This domain is very verbose
driver	Traces the channel driver	This domain is very verbose
threadlocal	Traces threads used to send SMS to the channel driver	This domain is very verbose

Note: If the “performance” domain is set to the DEBUG level, then all Oracle Database connections will be traced. In this case, a fine tuning on database access of XCT can be done (for example: execution plan, wait event). These traces are generated in the USER_DUMP_DEST directory defined in XCT’s `initSID.ora` database file.

Audit Trail

The Audit Trail facility provides services for generating a trace of the actions performed by the users of the system.

The management interface allows you to:

- Independently activate/deactivate the file/host mode.
- Set the temporary file location in file mode.
- Set the amount of allocated space for temporary file location in file mode.
- Activate/deactivate file storage mode, setting the storage location and the period that the temporary file must be archived in the storage location.
- Select either XML or CSV format for recording audit trail entries.

For details of how to configure audit trail generation, see “Configuring the Audit Trail” on page 14.

To determine whether to generate an audit trail for a particular product, refer to “Activating and Deactivating the Audit Trail” on page 32.

The Audit Trail file records information in either CSV (Comma Separator Value) or XML format.

CSV Format

Each audit trail file is made up of a single header line followed by zero or more lines, each containing an audit trail record. Lines are separated by a new line character (“\n”). Fields within each record are separated by a comma (“,”).

The header line has the following format:

AUDIT_TRAIL,creationDate

The format of each line is as follows:

productName,date,submodule,userID,function,result,freeText,objectType,objectID

The parameters are as follows:

Table 34 - Audit Trail Fields

Audit Trail Parameter	Description
<i>productName</i>	Product ID
<i>date</i>	Date and time of event in the format: YYYYMMDD HH:MM:SS
<i>submodule</i>	Name of module in which the event occurred (optional)
<i>userID</i>	User ID of the user that generated the event
<i>function</i>	Name of the function or service accessed by the user
<i>result</i>	Result of the action: OK, NOK, NOT_GRANTED
<i>freeText</i>	Free text associated with the event
<i>objectType</i>	Object type. Possible values are: MSISDN, Invocation ID, Profile name, Service name, Campaign ID, ICCID, Service implementation name, file name, Group ID, Vendor name, Definition name, Scenario label, and Scenario ID (optional)
<i>objectID</i>	Unique identifier of the object (optional)

Example

The following is an example of an audit trail in CSV format:

```
AUDIT_FILE,2002/03/21 09:26:05 CET
AccountRegistry,2002/03/21 09:26:05,AccessController,admgemalto,identify,OK,Password
AccountRegistry,2002/03/21 09:26:06,AccessController,admgemalto,grant,OK,
SAS1,2002/03/21 09:28:57,CI,gxsautouser,CISessionServerImpl:treatLogon,OK,
AccountRegistry,2002/03/21 09:31:16,AccessController,admgemalto,identify,OK,Password
AccountRegistry,2002/03/21 09:31:16,AccessController,admgemalto,grant,OK,
AccountRegistry,2002/03/21
09:35:04,AccessController,cca_richard,identify,OK,Password
AccountRegistry,2002/03/21 09:35:04,AccessController,cca_richard,grant,OK,
SAS1,2002/03/21 09:35:17,CI,gxsautouser,CISessionServerImpl:treatLogon,OK,
SAS1,2002/03/21 09:35:18,CI,gxsautouser,CISessionServerImpl:treatLogon,OK,
AccountRegistry,2002/03/21 09:35:54,AccessController,admgemalto,identify,OK,Password
AccountRegistry,2002/03/21 09:35:54,AccessController,admgemalto,grant,OK,
AccountRegistry,2002/03/21
09:36:56,AccessController,ccagcota,identify,NOK,DeniedAccess
AccountRegistry,2002/03/21
09:38:03,AccessController,cca_richard,identify,OK,Password
AccountRegistry,2002/03/21 09:38:03,AccessController,cca_richard,grant,OK,
SAS1,2002/03/21 09:47:05,CI,gxsautouser,CISessionServerImpl:treatLogout,OK,
```

Audit Trail DTD

The following DTD describes the XML format of audit trail files:

```
<!ELEMENT file ( header+, LINE* ) >
<!ELEMENT header ( title, date ) >
<!ELEMENT title ( #PCDATA ) >
<!ELEMENT date ( #PCDATA ) >

<!ELEMENT LINE ( freeText ) >
<!ATTLIST LINE result (OK|NOK|NOT_GRANTED) #REQUIRED >
<!ATTLIST LINE bsDate CDATA #REQUIRED >
<!ATTLIST LINE function NMTOKEN #REQUIRED >
<!ATTLIST LINE originator NMTOKEN #REQUIRED >
<!ATTLIST LINE submodule NMTOKEN #REQUIRED >
<!ATTLIST LINE userId NMTOKEN #REQUIRED >
<!ATTLIST LINE objectType NMTOKEN #REQUIRED >
<!ATTLIST LINE objectId NMTOKEN #REQUIRED >
<!ELEMENT freeText ( #PCDATA ) >
```

Billing Ticket Format

You can configure OTA Manager V5.1 to generate a billing ticket for each successful provisioning request or campaign. A typical billing ticket might contain the following information:

- A unique provisioning transaction identifier
- The subscriber identifier (MSISDN)
- The group to which the subscriber belongs
- The date and time of the transaction
- The issuer of the request
- The services that were provisioned by the transaction.

- The number of SMS messages that were sent to the subscriber.

Billing information can be:

- 1 Stored in system files in a specified folder on a hard disk
- 2 Exported for storage in an external billing system.

For details of how to configure billing ticket generation, see “Configuring Billing” on page 15.

To determine whether to generate billing tickets for a particular product, refer to “Activating and Deactivating Billing” on page 32.

Each billing ticket file is made up of a single header line followed by zero or more lines, each containing a billing record. Lines are separated by a new line character (“\n”). Fields within each record are separated by a comma (”,).

Each record has the following header format:

BILLING_FILE,VERSION_3,creationDate

The available types of record are:

- **SERVICE**
- **CMM_SERVICE**
- **CMM_SCENARIO**

Format of a line containing a SERVICE record:

*recordNumber,instanceName,date,invocationID,transID,**SERVICE**,requester,submitter,beneficiary,serviceName,status,server-specific-fields,invoker-specific-fields,bearerType,qualityService,direction,volume*

Format of a line containing a CMM_SERVICE record:

*recordNumber,instanceName,date,campaignID,transID,**CMM_SERVICE**,requester,submitter,beneficiary,serviceName,status,campaignName,scenarioName*

Format of a line containing a CMM_SCENARIO record:

*recordNumber,instanceName,date,campaignID,transID,**CMM_SCENARIO**,requester,submitter,beneficiary,scenarioName,status,campaignName*

where:

Table 35 - Billing Ticket Fields

Audit Trail Parameter	Description
<i>recordNumber</i>	The unique record number of the billing ticket entry
<i>instanceName</i>	The instance name of the product generating the billing ticket, for example “RCA1”
<i>date</i>	Date and time of event in the format: YYYYMMDD HH:MM:SS
<i>invocationID</i>	The unique identifier of the invocation.
<i>campaignID</i>	The unique campaign identifier of the campaign
<i>transID</i>	The unique identifier of the internal transaction (Invocation) performed by OTA Manager V5.1
<i>requester</i>	The user that initiated the request (for example a subscriber calling customer care requesting a particular service)
<i>submitter</i>	Identifies the person that sent the request. It corresponds to the account ID

Table 35 - Billing Ticket Fields (continued)

Audit Trail Parameter	Description
<i>beneficiary</i>	Identifies the request's target. It corresponds to the target MSISDN
<i>serviceName</i>	The service that was invoked
<i>status</i>	The result of the operation: FAILED or SUCCEEDED
<i>server-specific-fields</i>	Server-specific fields (comma separated)
<i>invoker-specific-fields</i>	Invoker-specific fields (comma separated)
<i>bearerType</i>	Type of the bearer. Possible values are SMS, WIR, or CAT-TP
<i>qualityService</i>	Requested quality of service
<i>direction</i>	MO or MT
<i>volume</i>	<p>The volume of data exchanged with the card.</p> <p>The value of this field depends on the transport type used to submit the request:</p> <ul style="list-style-type: none"> ■ For the SMS protocol, it contains the number of SMS messages exchanged ■ For the CAT-TP protocol, it contains the number of bytes exchanged
<i>completionStatus</i>	The completion status. Possible values are: FAILED, SUCCEEDED, OUT_OF_DATE or ABORTED
<i>campaignName</i>	The campaign that was invoked
<i>scenarioName</i>	The campaign scenario that was invoked

Example

The following is an example of billing tickets:

```
BILLING_FILE,VERSION_3,2009/01/09 11:40:40 CET
0,RCA1,20090109114040,,3211EBAFAB6E2,,SERVICE,admgemalto,admgemalto,060
2212123,ActivateADN,SUCCEEDED,,Billing 5.1,SMS,1,MT,1
1,RCA1,20090109114455,3211EBAFE9F97,,Service,admgemalto,admgemalto,060
2212123,ActivateFDN,FAILED,,Billing 09/01/09,SMS,,MT,0
2,RCA1,20090109115047,3211EBB03FB81,,Service,admgemalto,admgemalto,060
2212124,ActivateADN,SUCCEEDED,,ADN activation billi,SMS,1,MT,1
3,RCA1,20090109115312,3211EBB0633DA,,Service,admgemalto,admgemalto,060
2212125,ActivateADN,SUCCEEDED,,ADN activation,SMS,1,MT,1
```


SNMP Counters, Alarms, and Traps

This chapter describes the Management Information Base (MIB) file used by OTA Manager V5.1, and lists the counters and traps that are monitored by OTA Manager V5.1's Simple Network Management Protocol (SNMP) agent.

The MIB File

All items of information about OTA Manager V5.1 that can be monitored using SNMP are described in a hierarchical file called a Management Information Base (MIB).

An SNMP Manager can request and collect information from an SNMP agent's MIB file, as well as inspect the objects contained therein. For example, from the SNMP Manager you can examine the number of service requests that have been issued by users.

Note: OTA Manager V5.1 does not allow object values to be updated (with the **set** SNMP command) from the SNMP manager; values can only be read. However, an equivalent administrative command is available through the Core API to emulate the **set** SNMP command.

Each element manages specific objects, with each object having specific characteristics. Each object and characteristic has a unique object identifier (OID) consisting of numbers separated by decimal points (for example, 1.3.6.1.4.1.2682.1). These object identifiers naturally form a hierarchical tree structure. The MIB associates each OID with a readable label and various other parameters related to the object.

Here is an example of one of the OIDs used by the Framework (5496.2.20.1.3):

Table 36 - An Example OID

Code	Value
5496	gemplus node
2	OTA Manager node
20	platform status node
1	facilities node
3	local logging value

The above OID value has an associated text description - "Current state of the local Logging facility: 1 for OK - 0 for KO".

The SNMP Agent

OTA Manager V5.1 is able to host several kinds of products and allow each product type, through its SNMP agent, to add a product-specific part to the MIB. You are free to choose the content of the product-specific part of the MIB—the only constraint is that it must expand an existing node provided in the Master file.

Each time OTA Manager V5.1's SNMP agent starts, it generates a ".mib" file from the master file and any additional files that are present.

Master Files

A "master file" contains the MIB definition for the Framework. The name and location of this XML file is given in the **gxs.xml.file.location** property in the coreframework.ini file.

Any XML parsing errors in the master MIB file (incorrect XML structure, illegal nodes, and so on), stops both the parsing and the SNMP agent.

Additional Files

Additional files contain MIB definitions of hosted components or products. The location of these XML files is given in the **gxs.additional.file.location** property in the coreframework.ini file.

Additional files must comply with the `gc*mib.xml` naming convention. Any file located in the directory specified by the **gxs.additional.file.location** property that does not comply with this naming convention is ignored.

Note that if the **gxs.additional.file.location** property is missing, or references an incorrect location, no additional MIB files are processed. The SNMP agent is launched normally, but the generated MIB does not contain any additional parts. The detection of a parsing error on an additional XML MIB file (incorrect XML structure, illegal node, and so on) stops the parsing of the current file, but does not stop the SNMP agent.

DTD Format

The files containing MIB definitions must be in XML format and comply with the following DTD format:

```
<!ELEMENT DEFINITIONS ( IMPORTS, MIB ) >
<!ATTLIST DEFINITIONS root NMOKEN #REQUIRED >
<!ATTLIST DEFINITIONS oid NMOKEN #REQUIRED >
<!ELEMENT IMPORTS ( #PCDATA ) >
<!ELEMENT MIB ( GROUP ) >
<!ATTLIST MIB name NMOKEN #REQUIRED >
<!ATTLIST MIB oid NMOKEN #REQUIRED >
<!ELEMENT GROUP ( GROUP | OBJECT | TABLE | TRAP | VARBIND )* >
<!ATTLIST GROUP name ID #REQUIRED >
<!ATTLIST GROUP oid NMOKEN #REQUIRED >
<!ELEMENT OBJECT EMPTY >
<!ATTLIST OBJECT name NMOKEN #REQUIRED >
<!ATTLIST OBJECT access ( not-accessible | read-only ) #REQUIRED >
<!ATTLIST OBJECT description CDATA #IMPLIED >
<!ATTLIST OBJECT oid NMOKEN #REQUIRED >
<!ATTLIST OBJECT status ( mandatory | optional |
                           obsolete | deprecated ) #REQUIRED >
<!ATTLIST OBJECT syntax ( Counter | DisplayString |
                           INTEGER | TimeTicks| Gauge ) #REQUIRED >
<!ELEMENT TABLE ( ROW ) >
```

```

<!ATTLIST TABLE name NMTOKEN #REQUIRED >
<!ATTLIST TABLE access NMTOKEN #REQUIRED >
<!ATTLIST TABLE status NMTOKEN #REQUIRED >
<!ATTLIST TABLE entry NMTOKEN #REQUIRED >
<!ATTLIST TABLE oid NMTOKEN #REQUIRED >
<!ATTLIST TABLE description CDATA #IMPLIED >
<!ELEMENT ROW ( INDEX+, OBJECT+ ) >
<!ATTLIST ROW name NMTOKEN #REQUIRED >
<!ATTLIST ROW access NMTOKEN #REQUIRED >
<!ATTLIST ROW status NMTOKEN #REQUIRED >
<!ATTLIST ROW oid NMTOKEN #REQUIRED >
<!ATTLIST ROW description CDATA #IMPLIED >
<!ELEMENT INDEX EMPTY >
<!ATTLIST INDEX value NMTOKEN #REQUIRED >
<!ELEMENT TRAP ( VARIABLE+ ) >
<!ATTLIST TRAP name NMTOKEN #REQUIRED >
<!ATTLIST TRAP oid NMTOKEN #REQUIRED >
<!ATTLIST TRAP description CDATA #IMPLIED >
<!ELEMENT VARBIND ( OBJECT+ ) >
<!ELEMENT VARIABLE EMPTY >
<!ATTLIST VARIABLE name (ctxMessage | exceptionName | hostName |
                           orbAgentPort | senderName ) #REQUIRED >

```

SNMP Counters

For each product instance, OTA Manager V5.1 provides statistical information through the SNMP protocol. Counters can be included in simple leaves or tables.

Framework Counters

At the root location:

Table 37 - Framework Counters

Name	Type	Description
gxsDescriptor	String	The name of the OTA Manager V5.1 product instance
gxsUpTime	TimeTicks	Time elapsed since the platform was started
gxsComponentNb	Counter	Number of LinqUs Over-The-Air Suite products deployed

Under the **platformStatus** node:

Table 38 - platformStatus Counters

Name	Type	Description
localBilling	Integer	Current state of the local billing facility: 1 for OK - 0 for KO
remoteBilling	Integer	Current state of the remote billing facility: 1 for OK - 0 for KO
localLogging	Integer	Current state of the local logging facility: 1 for OK - 0 for KO
remoteLogging	Integer	Current state of the remote logging facility: 1 for OK - 0 for KO

Table 38 - platformStatus Counters

Name	Type	Description
localAuditTrail	Integer	Current state of the local trail facility: 1 for OK - 0 for KO
remoteAuditTrail	Integer	Current state of the remote trail facility: 1 for OK - 0 for KO
invocationRegistry	Integer	Current state of the Invocation registry facility: 1 for OK - 0 for KO
invocationRegistryDatabase	Integer	
fwkDatabase	Integer	Current state of the Framework database connection: 1 for OK - 0 for KO
auditTrailDatabase	Integer	Current state of the audit trail database: 1 for OK - 0 for Troubles
storageBilling	Integer	Current state of the storage mode of billing: 1 for OK - 0 for KO
storageLogging	Integer	Current state of the storage mode of logging: 1 for OK - 0 for KO
storageAuditTrail	Integer	Current state of the storage mode of audit trail: 1 for OK - 0 for KO
memoryBillingTotal	Integer	Total memory (in bytes) allocated to the JVM running the billing (this value is fixed during JVM life time)
memoryBillingFree	Integer	Current free memory (in bytes) of the JVM running the billing.
memoryLoggingTotal	Integer	Total memory (in bytes) allocated to the JVM running the logging (this value is fixed during the JVM's lifetime)
memoryLoggingFree	Integer	Current free memory (in bytes) of the JVM running the logging.
memoryAuditTrailTotal	Integer	Total memory (in bytes) allocated to the JVM running the audit trail (this value is fixed during the JVM's lifetime)
memoryAuditTrailFree	Integer	Current free memory (in bytes) of the JVM running the audit trail.
memoryInvocationRegistryTotal	Integer	Total memory (in bytes) allocated to the JVM running the invocation registry (this value is fixed during the JVM's lifetime).
memoryInvocationRegistryFree	Integer	Current free memory (in bytes) of the JVM running the invocation registry.
memoryCoreFrwkTotal	Integer	Total memory (in bytes) allocated to the JVM running the Core Framework (this value is fixed during the JVM's lifetime).
memoryCoreFrwkFree	Integer	Current free memory (in bytes) of the JVM running the Core Framework.
memoryCardMgrTotal	Integer	Total memory (in bytes) allocated to the JVM running the JVM running the Card Manager instance (this value is fixed during the JVM's lifetime).
memoryCardMgrFree	Integer	Current free memory (in bytes) of the JVM running the Card Manager instance.

UserAuthenticationTable

The **userAuthenticationTable** provides the number of authentications processed by the platform for every profile defined in the platform since the last restart.

Table 39 - The UserAuthentication Table

Name	Type	Description
authProfileTypeIndex	Integer	Type of the profile (administrator=0, customer care agent=1, subscriber=2)
authProfileTypeName	String	Name of the user profile (Administrator Profile, CCA Profile, Subscriber Profile)
authNb	Counter	Number of authentications

componentTable

The **componentTable** provides the product status (index **componentId**).

Table 40 - The componentTable

Name	Type	Description
orbAgentPort	Integer	The platform to which the product is connected
componentHost	String	Product location
componentName	String	Product name
componentStatus	Integer	Product status component Up/Down

The **componentStatus** checks the consistency between the expected status in the product life cycle and its real state. (For example, if the product should be in “kill” state and it is really killed, the status is “up”. If the product should be in “running” state and its process has been killed, the status is “down”.)

Card ManagerCounters

invocationManager Table

The **invocationManagerTable** provides information about the invocation manager using index **imIndex**.

Table 41 - The invocationManager Table

Name	Type	Description
imFrwkPort	Integer	ORB port number of the platform used by the invocation manager
imAddress	String	IP address of the machine on which the invocation manager is located
imName	String	Name of the product that hosts the invocation manager
imAcceptedInvocations	Counter	Number of accepted invocations
imRejectedInvocations	Counter	Number of rejected invocations
imSucceededInvocations	Counter	Number of succeeded invocations

Table 41 - The invocationManager Table (continued)

Name	Type	Description
imFailedInvocations	Counter	Number of failed invocations
imDeletedInvocations	Counter	Number of deleted invocations
imSuppressedInvocations	Counter	Number of suppressed invocations
imExpiredInvocations	Counter	Number of expired invocations.
imCreatedInvocationsHigh	Gauge	Number of created invocations (waiting for processing or scheduled) with urgent priority
imCreatedInvocationsNormal	Gauge	Estimated number of created invocations (waiting for processing or scheduled) with normal priority
imCreatedInvocationsLow	Gauge	Number of created invocations (waiting for processing or scheduled) with low priority
imInProcessInvocationsHigh	Gauge	Number of in process invocations with a High priority. Caution: The job which processes an invocation can be swapped in the database.
imInProcessInvocationsNormal	Gauge	Number of in process invocations with a Normal priority. Caution: The job which processes an invocation can be swapped in the database.
imInProcessInvocationsLow	Gauge	Number of in process invocations with a Low priority. Caution: The job which processes an invocation can be swapped in the database.

cattpModuleTable

The **cattpModule** table contains information about the CAT-TP protocol (index **cattpOnUdpMgrIndex**).

Table 42 - The cattpModule Table

Name	Type	Description
txSynPduNbr	Counter	The number of SYN PDU messages sent since the CAT-TP module was last started.
txSynAckPduNbr	Counter	The number of SYN ACK PDU messages sent since the CAT-TP module was last started.
txAckPduNbr	Counter	The number of ACK PDU messages sent since the CAT-TP module was last started.
txEackPduNbr	Counter	The number of EACK PDU messages sent since the CAT-TP module was last started.
txRstPduNbr	Counter	The number of RST PDU messages sent since the CAT-TP module was last started.
txRstAckPduNbr	Counter	The number of RST ACK PDU messages sent since the CAT-TP module was last started.
txNulPduNbr	Counter	The number of NUL PDU messages sent since the CAT-TP module was last started.
txNulAckPduNbr	Counter	The number of NUL ACK PDU messages sent since the CAT-TP module was last started.

Table 42 - The cattpModule Table (continued)

Name	Type	Description
rxSynPduNbr	Counter	The number of SYN PDU messages received since the CAT-TP module was last started.
rxSynAckPduNbr	Counter	The number of SYN ACK PDU messages received since the CAT-TP module was last started.
rxAckPduNbr	Counter	The number of ACK PDU messages received since the CAT-TP module was last started.
rxEackPduNbr	Counter	The number of EACK PDU messages received since the CAT-TP module was last started.
rxRstPduNbr	Counter	The number of RST PDU messages received since the CAT-TP module was last started.
rxRstAckPduNbr	Counter	The number of RST ACK PDU messages received since the CAT-TP module was last started.
rxNulPduNbr	Counter	The number of NUL PDU messages received since the CAT-TP module was last started.
rxNulAckPduNbr	Counter	The number of NUL ACK PDU messages received since the CAT-TP module was last started.
activePushConnectionsNbr	Counter	The number of currently open Push connections.
activePullConnectionsNbr	Counter	The number of currently open Pull connections.
rejectedConnectionsNbr	Counter	The number of connection requests that have been rejected since the UDP server last started.
rejectedPdusNbr	Counter	The number of PDUs (of all types except SYN PDU, which is included in the rejectedConnectionsNbr counter) that have been rejected since the UDP server last started.
badCheckSumPdusNbr	Counter	The number of PDUs with a bad checksum that have been rejected.

serviceTable

The **serviceTable** provides information by service name and product name. Indexes (**srvMgrIndex** and **srvIndex**) are used to retrieve a specific line in the SNMP protocol. In OTA Manager V5.1, the indexes do not contain information.

Table 43 - The Service Table

Name	Type	Description
svrMgrName	String	Name of the product where the service is declared
svrMgrAddress	String	IP address of the machine on which the product is running
svrFrwkPort	Integer	ORB port of the platform used by the product
svrName	String	Name of the service
svrInvocations	Counter	Number of successful svrName invocations by svrMgrName

channelDriverTable

The **channelDriverTable** provides information about the channel driver using the indexes **cDrvMgrIndex** and **cDrvIndex**.

Table 44 - The channelDriver Table

Name	Type	Description
cDrvMgrName	String	Name of the product where the channel driver is running
cDrvMgrAddress	String	IP address of the machine on which the product is running
cDrvFrwkPort	Integer	ORB port of the platform used by the product
cDrvName	String	Name of the channel driver
cDrvType	String	Type of the channel driver
cDrvVelocity	Integer	Velocity of the channel driver
cDrvRemotePort	Integer	Port of the SMSC connection
cDrvRemoteHost	String	Host name or IP address of the SMSC
cDrvSentMTs	Counter	Number of MT messages received by the channel driver. Note: Not the number of SMS messages sent to the SMSC.
cDrvDroppedMTs	Counter	Number of MTs dropped (messages that cannot be sent to the SMSC)
cDrvReceivedMOs	Counter	Number of Mobile Originated (MO) messages received
cDrvAckSMSCsOK	Counter	Number of OK acknowledgements received from the SMSC (MT)
cDrvAckSMSCsFailed	Counter	Number of failed acknowledgements received from the SMSC (MT)
cDrvAckSIMsOK	Counter	Number of OK acknowledgements received from SIM card (MT)
cDrvAckSIMsFailed	Counter	Number of failed acknowledgements received from SIM card (MT)
cDrvAckMOsOK	Counter	Number of OK acknowledgements sent (MO)
cDrvAckMOsFailed	Counter	Number of failed acknowledgement sent (MO)
cDrvActivationStatus	Integer	True if the channel driver is activated
cDrvConnectionStatus	Integer	True if the channel driver is connected
cDrvReceivedAlerts	Counter	Number of Received alerts
cDrvDisconnectNumber	Integer	Number of disconnections that have occurred since the last activation time for a specific channel driver (identified by its index in the table)
cDrvDisconnectTime	Integer	Total connection downtime time (in seconds) for a specific channel driver (identified by its index in the table) since last activation time.

storageAreaTable

The **storageAreaTable** provides a list of storage area entries. The number of entries is equal to the number of active products (indexes **saMgrIndex**, **saIndex**).

Table 45 - The storageArea Table

Name	Type	Description
saMgrName	String	The product where the storage area is instantiated
saName	String	The storage area name
saType	String	The storage area type
saSize	Gauge	The number of objects stored in the storage area
salns	Counter	The number of insertions
saSup	Counter	The number of deletions

cardMgrTable

The **cardMgrTable** provides the status specific to the Card Manager (Index **cardMgrId**).

Table 46 - The cardMgr Table

Name	Type	Description
orbAgentPort	Integer	The platform to which the product is connected
cardMgrAddress	String	Product location
cardMgrName	String	Card Manager name
cardMgrStatus	Integer	Card Manager status: 1 for OK, 0 KO. See cardManagerTrouble trap.
securityDatabase	Integer	Security Settings database status
cardDatabase	Integer	Card Manager database status
formattingLayer	Integer	Formatting layer status formattingLayer Down
cardServiceLayer	Integer	Card service layer status cardServiceLayer Up/Down
smscDriverMgr	Integer	SMSC driver manager status: 1 for OK, 0 KO. See smscDriverManagerDown trap.
smscDriver	Integer	The connection between SMSC manager and drivers: 1 for OK, 0 KO. See smscDriverDown trap.
smsc	Integer	SMSC status smscError
cattpMgr	Integer	
memoryCardMgrTotal	Integer	The total amount of VM memory allocated to running the component.
memoryCardMgrFree	Integer	Current amount of free VM memory.

X15CmMgrTable

The **X15CmMgrTable** provides information about the XCT campaign engine (index **X15ModuleId**).

Table 47 - The X15McMgr Table

Name	Type	Description
X15ModuleId	Integer	The unique identifier in OTA Manager V5.1 Framework
PreThroughput	Integer	Current invocation throughput (in invocations per second) for the XCT pre-processing module
PreCounter	Integer	Current invocation counter for the XCT pre-processing module
PostThroughput	Integer	Current invocation throughput (in invocations per second) for the XCT post-processing module.
PostCounter	Integer	Current invocation counter for the XCT post-processing module
SubmitThroughput	Integer	Current throughput (expressed in SMS per second) for the XCT sending module (excluding Inquiry commands)
SubmitCounter	Counter	Current submit counter for the XCT sending module
DSThroughput	Integer	Current throughput on Delivery Status reception
DSCounter	Counter	Current Delivery Status counter for the XCT sending module
QueryThroughput	Integer	Current throughput on query
QueryCounter	Counter	Current query counter for the XCT sending module
MOThroughput	Integer	Current throughput on MO reception
MOCounter	Counter	Current MO counter for the XCT sending module
AlertsThroughput	Integer	Current throughput on SS7 alerts
AlertsReceived	Counter	Current alerts counter for the XCT sending module
InvocationCacheHit	Integer	Cache hit for invocation cache
SMSCacheHit	Integer	Cache hit for SMS cache
SendFlowControlTank	Integer	State of the flow control tank (of the sending module)
SmscConnectionState	Integer	State of the SMSC connection [0=NOT connected, 1=connected]
X15DatabaseState	Integer	State of the XCT database [0=NOT connected, 1=connected]

Campaign Manager Counters

The following counters reflect the state of campaigns:

Table 48 - Campaign Counters

Counter Name	Description
campaigns	Total number of campaigns declared
createdCampaigns	Number of created campaigns declared
pausedCampaigns	Number of paused campaigns declared

Table 48 - Campaign Counters (continued)

Counter Name	Description
frozenCampaigns	Number of frozen campaigns declared
abortedCampaigns	Number of aborted campaigns declared
idleCampaigns	Number of idle campaigns declared
executingCampaigns	Number of executing campaigns declared
terminatingCampaigns	Number of terminating campaigns declared
terminatedCampaigns	Number of terminated campaigns declared

Target Counters

The following counters reflect the state of a single campaign on a set of targets:

Table 49 - Target Counters

Counter Name	Description
campaignID	Campaign identifier
campaignName	Name of the campaign
totalTargets	Total number of targets in the campaign
scheduledTargets	Number of targets on which the scenario execution has not yet begun
inProcessTargets	Number of targets on which the scenario is currently being processed
notYetCreatedTargets	Number of targets still to be created
inRetryTargets	Number of targets on which one or more unitary task is being retried
terminatedSucceededTargets	Number of targets on which the scenario terminated successfully
terminatedNotSucceededTargets	Number of targets on which the scenario terminated unsuccessfully
outOfDateTargets	Number of targets on which the scenario execution was not completed in the allowed time
abortedTargets	Number of targets on which the scenario execution was not completed before the campaign was aborted

Traffic Flow Counters

The following counters provide information about connected products:

Table 50 - Traffic Flow Counters

Counter Name	Description
abstractSEEMaxThroughput	The maximum request throughput to the product, in requests per second
abstractSEEMaxCongestion	The maximum number of requests that the product can process
abstractSEEThroughput	The current request throughput to the product, in requests per seconds

Table 50 - Traffic Flow Counters

Counter Name	Description
abstractSEENName	Name of the connected product
abstractSEEState	Whether the product is CONNECTED or DISCONNECTED
abstractSEECongestion	The current number of requests submitted to the connected product.

SNMP Traps

Traps are messages generated by OTA Manager V5.1's SNMP agent and sent to SNMP management software to report alarms. This proactively notifies the SNMP manager as soon as an alarm condition occurs, instead of waiting for the SNMP manager to query the product's status.

Traps can be generated by both the Framework (billing, logging, audit trail, invocation registry, database server) and individual services (SMS channel manager, Card Manager, Campaign Manager, and so on).

Each trap contains the following information:

- Code: uniquely identifies the alarm.
- Timestamp: time that the alarm was generated.
- Severity level: (critical, major, minor, warning, cancel).
- Sender Name: name of the sender.
- Description: contextual message (if available).

In addition, each trap can contain the following variable information:

- Sender name: the module or product that detects the alarm.
- Address agent: contains the address of the host machine that physically sends the traps (this is always the machine that hosts the Core Framework). The IP address resolution mechanism can be overridden by the Java property **snmp.source.address.override**. This property can be set in the Core Framework command line (add it to the COREFRAMEWORK_CUSTOM_OPTIONS variable in the `gemconnect.custom` file). If it is set, the **Address agent** field of the trap contains the IP address produced by the DNS resolution of the **snmp.source.address.override** host. This is useful when the machine that hosts the Framework has two network interfaces. The VIP case is managed, as DNS resolution is performed each time a trap is sent.

Note: In the case of a cluster, you cannot use a fixed host name because the address would be wrong if the agent is migrated to a different host.

- Host name: the name or address of the host machine. In the case of a VIP address, the real address is returned. Host names can be overridden by the Java property **snmp.hostname.override**. This property can be set on each Framework module's Java command line (BS_COMPONENT table of the Framework), or the Core Framework command line (add it to the COREFRAMEWORK_CUSTOM_OPTIONS variable in the `gemconnect.custom` file). If it is set, the **Host name** field of the trap contains the IP address produced by the DNS resolution of the **snmp.hostname.override** host.

Note: In the case of a cluster, you cannot use a fixed host name because the address would be wrong if the product is migrated to another host.

- ORB agent port (for facility alarms, this uniquely identifies the platform).
- The component name, for example, “GCCM” for the Campaign Manager.
- Exception name: the name of the exception that raised the trap.
- Contextual message: information to help analyze and solve the problem; this message may be empty if previous fields contain all the necessary information.

Note that no severity escalation is employed: one trap is generated to notify the management software of service unavailability, and another trap for renewed service availability.

Framework Traps

Each time a trap is sent, a corresponding counter may exist the value of which is updated to represent the current component or feature status. The value is set to 1 if the service is functioning normally, and it is set to 0 to indicate that a warning trap (“down” or “trouble”) has been sent.

Traps available under the **gxsFacility** node:

Table 51 - gxsFacility Traps

Trap Name	Description	Associated counter
localBillingDown	Billing on local server is unavailable	localBilling
localBillingUp	Billing on local server is restored	localBilling
remoteBillingDown	Billing on remote server is unavailable	remoteBilling
remoteBillingUp	Billing on remote server is restored	remoteBilling
localLoggingDown	Logging on local server is unavailable	localLogging
localLoggingUp	Logging on local server is restored	localLogging
remoteLoggingDown	Logging on remote server is unavailable	remoteLogging
remoteLoggingUp	Logging on remote server is restored	remoteLogging
localAuditTrailDown	Audit Trail on local server is unavailable	localAuditTrail
localAuditTrailUp	Audit Trail on local server is restored	localAuditTrail
remoteAuditTrailDown	Audit Trail on remote server is unavailable	remoteAuditTrail
remoteAuditTrailUp	Audit Trail on remote server is restored	remoteAuditTrail
invocationRegistryDatabase Trouble	Invocation Registry database trouble	invocationRegistry
invocationRegistryDatabase Up	Invocation Registry database recovered	invocationRegistry
databaseServerDown	Problem of connection with database	fwkDatabase
databaseServerUp	Connection with database restored	fwkDatabase
auditTrailDatabaseTrouble	Problem with Audit Trail database	auditTrailDatabase
auditTrailDatabaseUp	Audit trail database OK	auditTrailDatabase
storageBillingDown	Storage mode of Billing is unavailable	storageBilling
storageBillingUp	Storage mode of Billing is restored	storageBilling
storageLoggingDown	Storage mode of Logging is unavailable	storageLogging

Table 51 - gxsFacility Traps (continued)

Trap Name	Description	Associated counter
storageLoggingUp	Storage mode of Logging is restored	storageLogging
storageAuditTrailDown	Storage mode of Audit Trail is unavailable	storageAuditTrail
storageAuditTrailUp	Storage mode of Audit Trail is restored	storageAuditTrail

Traps available under the **gxsComponentState** node:

Table 52 - gxsComponent Traps

Trap Name	Description	Associated Counter
componentDown	Used by the base system to notify an unrequested state change of product	componentStatus
componentUp	Used by the base system to notify that a product is ready (connection restored)	componentStatus

Traps available under the **gxsLifeCycle** node:

Table 53 - gxsLifeCycle Traps

Trap Name	Description	Associated Counter
gxsStart	Raised at the start of OTA Manager V5.1	N/A
gxsStop	Raised at the stop of OTA Manager V5.1	N/A

Card Manager Traps

Traps available under the **gxsCardManager** node:

Table 54 - gxsCardManager Traps

Trap Name	Description	Associated Counter
securityDatabaseTrouble	Connection to security database is lost	securityDatabase
securityDatabaseUp	Connection to security database recovered	securityDatabase
formattingLayerDown	Trouble with bad formatting or library class	formattingLayer
cardDatabaseTrouble	Connection to the Framework database was lost	cardDatabase
cardDatabaseUp	Connection to the Framework database recovered	cardDatabase
cardServiceLayerDown	Invalid or missing parameter's file for plug-in	cardServiceLayer
cardServiceLayerUp	File has been added or corrected	cardServiceLayer
cardManagerTrouble	Card Manager unavailable for database problem.	cardMgrStatus
cattpMgrUp	Connection made with CAT-TP.	N/A
cattpMgrDown	Connection with CAT-TP lost.	N/A

Traps available under the **gxsSMSChannelManager** node:

Table 55 - gxsChannelManager Traps

Trap Name	Description	Associated counter
smscDriverDown	Connection lost between a driver and SMSC	smscDriver
smscDriverUp	Connection between a driver and SMSC restored	smscDriver
smscDriverManagerDown	Connection to database lost	smscDriverMgr

Traps available under the **gxsSMSC** node:

Table 56 - gxsSMSC Traps

Trap Name	Description	Associated counter
smscError	Specific SMSC error	N/A

Traps available under the **gxsSSAComponent** node:

Table 57 - gxsCampaignManager Traps

Trap Name	Description	Action	Severity	Associated counter
ssaEndOfAlert	Raised when the SSA size is now under the critical limit	This is a clear alarm. No action is required.	Clear	N/A
ssaAlert	Raised when the SSA size is now above the critical limit, or the connection to the database is lost.	Increase the size of the SSA database. Use <i>classicalCampaigns TuningWizard.xls</i> for help.	Major	N/A
	Raised when the connection to the database is lost.	Check the database state.	Major	N/A

XCT Traps

Traps available under the **X15CmMgr** node:

Table 58 - X15CmMgr Traps

Trap Name	Description	Action	Severity	Associated counter
X15DatabaseUp	XCT database connection is reconnected	This is a clear alarm: no action is required.	Clear	N/A
X15DatabaseDown	XCT database connection is lost	Check the database state	Major	N/A
X15SmscUp	XCT SMSC connection is reconnected	This is a clear alarm: no action is required.	Clear	N/A
X15SmscDown	XCT SMSC connection is lost	Check the connection between XCT and the SMSC	Minor	N/A

Local Billing Facility (File Mode)

Table 59 - Local Billing Facility

Trap Name	Cause	Variable Binding	Action	Severity
localBillingDown	File system full (I/O exception)	Sender name (facility name) Host name ORB agent port (uniquely identifies the framework if two platforms are running on the same host) Exception name Contextual message (error message)	The file system is full. Delete the old or unused files to free more space on the disk. You must check the remaining disk space regularly. Cyclical files are used for billing, so the billing functionality is not responsible for loss of space on the disk. If no action is taken, all new billing tickets will be lost. To check the configuration: 1. Log on to the OTA Manager V5.1 interface as an administrator and click Platform Management . 2. Click Platform config. > Billing . Check that File mode or Storage mode is enabled.	Major
localBillingUp	Back to normal state: all is OK	Sender name (facility name) Host name ORB agent port (uniquely identifies the framework if two platforms are running on the same host) Contextual message	This is a clear alarm. No action is required.	Clear

Local Billing Facility (Storage Mode)

Table 60 - Local Billing Facility

Trap Name	Cause	Variable Binding	Action	Severity
storageBillingDown	File system full (I/O exception)	Sender name (facility name) Host name ORB agent port (uniquely identifies the Framework if two platforms are running on the same host) Exception name Contextual message (error message)	The file system is full. Delete the old or unused files to free more space on the disk. You must check the remaining disk space regularly. Cyclical files are used for billing, so the billing functionality is not responsible for loss of space on the disk. If no action is taken, all new billing tickets will be lost. To check the configuration: 1. Log on to the OTA Manager V5.1 interface as an administrator and click Platform Management . 2. Click Platform config. > Billing . Check that File mode or Storage mode is enabled.	Major
storageBillingUp	Back to normal state: all is OK	Sender name (facility name) Host name ORB agent port (uniquely identifies the Framework if two platforms are running on the same host) Contextual message	This is a clear alarm. No action is required.	Clear

SNMP Counters, Alarms, and Traps

Remote Billing Facility

Table 61 - Remote Billing Facility

Trap Name	Cause	Variable Binding	Action	Severity
remoteBillingDown	TCP connection lost	Sender name (facility name) Host name ORB agent port (uniquely identifies the framework if two platforms are running on the same host) Exception name Contextual message (error message)	In remote mode, billing tickets are sent over a TCP/IP connection to a custom program. If the TCP/IP connection is lost, check the network equipment first. For example, you can use a ping command to check if the host, where the client program runs, is reachable. If not, the network is partially down. Next, check the client application. Consult the client application documentation to determine if you need to restart the custom application.	Major
remoteBillingUp	Back to normal state: all is OK	Sender name (facility name) Host name ORB agent port (uniquely identifies the framework if two platforms are running on the same host) Contextual message	To check the configuration: 1. Log on to the OTA Manager V5.1 interface as an administrator and click Platform management 2. Click Platform config. > Billing . Check the Host mode.	Clear

Local Logging Facility

Table 62 - Local Logging Facility

Trap Name	Cause	Variable Binding	Action	Severity
localLoggingDown	File system full (IO exception)	Sender name (facility name) Host name ORB agent port (uniquely identifies the framework if two platforms are running on the same host) Exception name Contextual message (error message)	The file system is full. Delete the old or unused files to free more space on the disk. You must check the remaining disk space regularly. Cyclic files are used for logging, so the billing functionality is not responsible for loss of space on the disk. If no action is taken, all new logging entries will be lost. To check the configuration: 1. Log on to the OTA Manager V5.1 interface as an administrator and click Platform management 2. Click Platform config. > Logging . Check that the File mode or Storage mode is enabled.	Minor
localLoggingUp	Back to normal state: all is OK	Sender name (facility name) Host name ORB agent port (uniquely identifies the framework if two platforms are running on the same host) Contextual message	This is a clear alarm. No action is required.	Clear

Remote Logging Facility

Table 63 - Remote Logging Facility

Trap Name	Cause	Variable Binding	Action	Severity
remoteLoggingDown	TCP connection lost	Sender name (facility name) Host name ORB agent port (uniquely identifies the framework if two platforms are running on the same host) Exception name Contextual message (error message)	In remote mode, logging entries are sent over a TCP connection to a custom program. If the TCP connection is lost, check the network equipment first. For example, you can use a ping command to check if the host, where the client program runs, is reachable. If not, the network is partially down. Next, check the client application. Consult the client application documentation to determine if you need to restart the custom application. To check the configuration: 1. Log on to the OTA Manager V5.1 interface as an administrator and click Platform management 2. Click Platform config. > Logging . Check the Remote mode.	Minor
remoteLoggingUp	Back to normal state: all is OK	Sender name (facility name) Host name ORB agent port (uniquely identifies the framework if two platforms are running on the same host) Contextual message	This is a clear alarm. No action is required.	Clear

Local Audit Trail Facility

Table 64 - Local Audit Trail Facility

Trap Name	Cause	Variable Binding	Action	Severity
localAuditDown	File system full (I/O exception)	Sender name (facility name) Host name ORB agent port (uniquely identifies the framework if two platforms are running on the same host) Exception name Contextual message (error message)	The file system is full. Delete the old or unused files to free more space on the disk. You must check the remaining disk space regularly. Cyclic files are used for the audit trail, so the audit trail functionality is not responsible for loss of space on the disk. If no action is taken, all new audit trails will be lost. To check the configuration: 1. Log on to the OTA Manager V5.1 interface as an administrator and click Platform management 2. Click Platform config. > Audit trail . Check that File mode or Storage mode is enabled.	Major
localAuditUp	Back to normal state: all is OK	Sender name (facility name) Host name ORB agent port (uniquely identifies the Framework if two platforms are running on the same host) Contextual message	This is a clear alarm. No action is required.	Clear

Remote Audit Trail Facility

Table 65 - Remote Audit Trail Facility

Trap Name	Cause	Variable Binding	Action	Severity
remoteAuditDown	TCP connection lost	Sender name (facility name) Host name ORB agent port (uniquely identifies the framework if two platforms are running on the same host) Exception name Contextual message (error message)	In remote mode, audit trails are sent over a TCP/IP connection to a custom program. If the TCP connection is lost, check the network equipment first. For example, you can use a ping command to check if the host, where the client program runs, is reachable. If not, the network is partially down. Next, check the client application. Consult the client application documentation to determine if you need to restart the custom application. To check the configuration: 1. Log on to the OTA Manager V5.1 interface as an administrator and click Platform management 2. Click Platform config. > Audit trail . Check the Remote mode.	Major
remoteAuditUp	Back to normal state: all is OK	Sender name (facility name) Host name ORB agent port (uniquely identifies the framework if two platforms are running on the same host) Contextual message	This is a clear alarm. No action is required.	Clear

Invocation Registry Facility

Table 66 - Invocation Registry Facility

Trap Name	Cause	Variable Binding	Action	Severity
invocationRegistryDatabaseTrouble	Database connection lost No more resources	Sender name (facility name) Host name ORB agent port (uniquely identifies the framework if two platforms are running on the same host) Exception name Contextual message	<p>A possible cause is loss of the database connection. Using the information in the contextual message specified in the trap varbinds, check the database or the network.</p> <p>There are several possible causes: no more space in the database file system, the maximum number of Oracle Database processes is exceeded, or other database-critical resources are missing. If the problem involves the database, use the contextual message to determine the cause and repair the problem. Restarting the database may be necessary. If the problem is the network, check the network equipment to determine the cause.</p> <p>In rare cases, the cause is just a single SQL Oracle Database exception, such as a value too long for the insertion. In this case, this event must be logged, but it can be acknowledged and ignored.</p> <p>For all other cases of a lost connection, the invocation registry module must be restarted. This module is not completely database loss proof. Do not forget to check the invocation registry parameter if the problem occurs when the module is starting.</p> <p>To check the configuration:</p> <ol style="list-style-type: none"> 1. Log on to the OTA Manager V5.1 interface as an administrator and click Platform management. 2. Click Registry. 	Major

Database Server Facility

Table 67 - Database Server Facility

Trap Name	Cause	Variable Binding	Action	Severity
databaseServerDown	Database connection lost	Sender name (facility name) Host name ORB agent port (uniquely identifies the framework if two platforms are running on the same host) Exception name Contextual message	A possible cause is a lost database connection. Using the information in the contextual message specified in the trap varbinds, check the database or the network. There are several possible causes: no more space in the database file system, the maximum number of Oracle processes is exceeded, or other database-critical resources are missing. If the problem involves the database, use the contextual message to determine the cause and repair the problem. Restarting the database may be necessary. If the problem is the network, check the network equipment to determine the cause. In rare cases, the cause is just a single SQL Oracle exception, such as a value too long for the insertion. In this case, this event must be logged, but it can be acknowledged and ignored. For all other cases of a lost connection, the invocation registry module must be restarted. This module is not completely database loss proof. Do not forget to check the invocation registry parameter if the problem occurs when the module is starting. The file <code>coreframework.ini</code> (by default in the <code>config</code> directory of the OTA Manager V5.1 platform installation) defines the database parameter. Check this file if the problem occurs when the platform is starting.	Major
databaseServerUp	Connection to database restored. However, although the platform is able to restore the connection to the database, it is not in a stable state; OTA Manager V5.1 must still be restarted after a connection is lost to the database.	Sender name (facility name) Host name ORB agent port (uniquely identifies the framework if two platforms are running on the same host) Contextual message	This a clear alarm. No action is required.	Clear

Note: Although the OTA Manager V5.1 Framework can restore a connection to a database, the state is not stable. OTA Manager V5.1 must be restarted after losing a connection to a database.

Processor CrashObserver Facility (Internal Facility)

Table 68 - Database Crash Observer Facility (Internal Facility)

Trap Name	Cause	Variable Binding	Action	Severity
componentDown	Sent by the Framework to indicate that there is an unrequested change of state for the product (product process killed without using the OTA Manager V5.1 API)	Sender name (facility name) Host name ORB agent port (uniquely identifies the framework if two platforms are running on the same host) Exception name Contextual message	This alarm is sent when the processCrashObserver detects an abnormal ending of a component such as a functional facility or product. If the product or component does not restart automatically, you must restart it manually.	Major
componentUp	Sent by the Framework to indicate that the product is ready (connection restored)	Sender name (facility name) Host name ORB agent port (uniquely identifies the framework if two platforms are running on the same host) Exception name Contextual message	This is a clear alarm. No action is required.	Clear

Service Traps

SMSC Channel Manager Service

Table 69 - SMSC Channel Manager Service

Trap Name	Cause	Variable Binding	Action	Severity
smscDriverDown	Connection between a driver and SMSC lost	Sender name (product name) Host name ORB agent port Driver identifier Exception name Contextual message	First check the network. If the problem is caused by the SMSC, the SMSC must be restarted to restore the connection. The OTA Manager platform will not receive anything through this driver channel until the connection is restored. Depending on the class of the driver, the severity of this alarm may be minor. To check the driver configuration: <ol style="list-style-type: none">1. Log on to OTA Manager V5.1 as an administrator2. Click Platform management then click Product configuration.3. Select the corresponding product, and click Product Interface.4. In the Driver Management area, click Manage and update the driver configuration if necessary.	Major
smscDriverUp	Connection between a driver and SMSC restored	Sender name (product name) Host name ORB agent port Driver identifier Contextual message	This is a clear alarm. No action is required.	Clear
smscDriverManagerDown	Lost connection to database (impossible to add or update a driver)	Sender name (product name) Host name ORB agent port Exception name Contextual message	You cannot add, delete, or update a driver until the connection to the database is restored. The product must be restarted after the database connection is restored. Check the product parameters if the problem occurs when the product is starting. Parameter group: GENERAL Product parameters: DB_URL, DB_USER, DB_PSWD and JDBC_DRIVER	Minor

Card Manager Service

Table 70 - Card Manager Service

Trap Name	Cause	Variable Binding	Action	Severity
securityDatabaseTrouble	Connection to security database lost	Sender name (product name) Host name ORB agent port Exception name Contextual message	This alarm is sent when the Message Builder (Card Manager module) loses its connection to the security database. Using the information in the contextual message, check the network first, then the database, and finally the configuration of the product parameters. Parameter group: MESSAGE_BUILDER Product parameters: DB_URL, DB_USER, DB_PSWD and JDBC_DRIVER	Major
formattingLayerDown	Formatting library or class incorrect	Sender name (product name) Host name ORB agent port Exception name Contextual message	The Message Builder initialization failed. The GSM 03.48 format and unformat processes will not work. You must enable the logging trace: 1. Log on to OTA Manager V5.1 as an administrator. 2. Click Platform management and click Product configuration . 3. Click Edit trace level and select the maximum level of trace for the "MB_general" domain and sub-domains and restart the product to obtain more information. The problem is usually the result of incorrect configuration of the Card Manager PlugIn (incorrect CLASSPATH or PATH). The Message Builder is a mandatory module of the Card Manager product. This alarm is usually thrown after an incorrect installation or immediately following an update or an incorrect migration.	Major
cardDatabaseTrouble	Lost connection to card Framework database	Sender name (product name) Host name ORB agent port Exception name Contextual message	This alarm is sent when the Card Manager loses its connection to the database. Using the information in the contextual message, check the network first, then the database, and finally the configuration of the product parameters. Parameter group: GENERAL Product parameters: DB_URL, DB_USER, DB_PSWD and JDBC_DRIVER	Major

Table 70 - Card Manager Service (continued)

Trap Name	Cause	Variable Binding	Action	Severity
cardServiceLayerDown	Invalid or missing parameters file for service plug-in	Sender name (product name) Host name ORB agent port Service name Exception name Contextual message	This alarm is usually thrown after an incorrect configuration of the Card Manager product. 1. Log on to the OTA Manager V5.1 interface as an administrator and click Platform management . 2. Select Product configuration , then select a product, and click Configure . 3. On the Product configuration window, click Product Interface . 4. Select Service Management or Service Implementation , and use the information in the contextual message, the platform log, and the product manual to fix the configuration problem. If this event is sent without any modification, the configuration files have probably been altered (files deleted by mistake, for example). The event can be considered as "Major" in this case because the Card Manager product will not work correctly until the problem is fixed.	Major/Minor
cardServiceLayerUp	File has been added or corrected	Sender name (product name) Host name ORB agent port Service name Contextual message	This is a clear alarm. No action is required	Clear

Table 70 - Card Manager Service (continued)

Trap Name	Cause	Variable Binding	Action	Severity
cardManagerTrouble	Card Manager unavailable because of database problems (for example, a problem with a tablespace definition in the database)	Sender name (product name) Host name ORB agent port Exception name Contextual message	This alarm is sent when the card manager loses its connection to the database. It is the same as the cardDatabaseTrouble except it occurs when the product is starting. Using the information in the contextual message, check the network first, then the database, and finally the configuration of the product parameters. Parameter group: GENERAL Product parameter: DB_URL, DB_USER, DB_PSWD and JDBC_DRIVER.	Major
cattpMgrUp	CAT-TP module is starting up	Sender name (product name) Host name ORB agent port Exception name Contextual message	This is a clear alarm. No action is required.	Clear
cattpMgrDown	CAT-TP module is not started.	Sender name (product name) Host name ORB agent port Exception name Contextual message	The CAT-TP module is not started. As the CAT-TP module is optional, first check that the Card Manager product parameters relating to this module are set correctly.	Major/Minor

Note: Due to the Card Manager design, when one card service goes down, most services will be unavailable.

Campaign Manager Service

Table 71 - Campaign Manager Service Traps

Trap Name	Cause	Variable Binding	Action	Severity
databaseDown	Lost connection to database	Sender name (product name) Host name ORB agent port Component name: "GCCM" Exception name Contextual message	The problem may be a result of: 1. Network problem: lost link between Campaign Manager and its database - Check the network connections 2. Database crash - Restart the database.	Major
databaseUp	Restored database connection	Sender name (product name) Host name ORB agent port Component name: "GCCM" Contextual message	This is a clear alarm. No action is required.	Clear

Other Traps

SMSC

The SMSC trap is not mapped to a facility or service.

Table 72 - SMSC Error Traps

Trap Name	Cause	Variable Binding	Action	Severity
smscError	Specific SMSC error	Sender name (product name) Host name ORB agent port Driver identifier (the driver that receives the error) Exception name Contextual message (SMSC error message)	This is a special trap that is sent when SMS drivers receive an error acknowledgement from the SMSC. It indicates either a problem with the SMSC itself, or a message that is formatted incorrectly. The events must be logged, and according to the SMSC error code, the support team can be contacted for further investigation.	Minor

Error Messages

OTA Manager V5.1 has a number of components that provide error messages, as shown in “Table 73”. These error messages are copied into the Base System database when the product is installed.

Table 73 - Components Generating Error Messages

Type	Description
GENERAL	General OTA Manager V5.1 errors, including database errors
BS	Base System errors
CCI	Common Communication Interface (CCI) messages used in OTA Manager V5.1 products for exceptions (CIEception) and responses (ResponseHandler)
RCA	Card Manager errors

General Error Messages

Table 74 - General Error Messages

Error Code	Meaning
GENERAL-0	No Error
GENERAL-1	System Error. See detailed message for more explanation.
GENERAL-2	Unable to access a Base System Functional module. Check the Base System activity
GENERAL-3	Unable to initialize the logging. See detailed message for more explanation
GENERAL-4	Unable to read a Base System parameter
GENERAL-5	Unable to initialize a component
GENERAL-6	Transition not allowed
GENERAL-7	Request with incorrect product ID
GENERAL-8	Transition failed
GENERAL-9	Service not found
GENERAL-10	The request contains more than one service
GENERAL-11	Service not allowed for the current user

Table 74 - General Error Messages (continued)

Error Code	Meaning
GENERAL-12	Deactivated service
GENERAL-13	Component instantiation error
GENERAL-14	Unknown parameters
GENERAL-15	Mandatory parameter missing
GENERAL-16	Invalid parameter format
GENERAL-17	Invalid request format
GENERAL-18	Object already exists in database
GENERAL-19	Object not found
GENERAL-20	Deactivated component
GENERAL-21	Invalid XML definition
GENERAL-22	XML parsing error
GENERAL-23	Request without service
GENERAL-24	Request without destination
GENERAL-25	Request without verb
GENERAL-100	Data not found in database
GENERAL-101	Database internal error
GENERAL-102	Invalid request
GENERAL-103	Invalid data format
GENERAL-104	Database constraint violation
GENERAL-105	Data out of range
GENERAL-106	Concurrency error
GENERAL-150	No ID
GENERAL-666	Undefined Error. See detailed message for more explanation.
GENERAL-1013	Server unavailable
GENERAL-2000	The checking message is not successful

Base System Error Messages

com.gemplus.gxs.api.exception Package

The `com.gemplus.gxs.api.exception` package throws the exceptions shown in the following table:

Table 75 - Base System Error Messages

Java Class	Meaning
AlreadyExistsException	The object is already defined in the database.
AlreadyUsedException	The object is already used or defined in the platform.
NoMatchingObjectException	The object you want to use or modify does not exist in the OTA Manager platform.
DeniedAccessException	You are not allowed to invoke this method.
InvalidStateException	Your action is incompatible with the current state of the object.
InvocationNotFoundException	The invocation is no longer in the registry.
InvalidCursorException	You are trying to use a cursor that is already closed, or that is impossible to build.
NoMoreResourceException	The registry has no more resources for processing.
InvalidInvocationException	The invocation is not valid.
MalformedSearchCriteriaException	The syntax of the search criteria is incorrect.
ViolatedDependencyException	The profile you want to delete is associated with existing accounts.
ChainAlreadyDefinedException	The chain is already defined.
NoMatchingChainException	There is no matching chain.
NoMatchingListenerException	There is no matching listener.
ExpiredAccountException	The account expiration date has been reached.
TimeoutException	The token validity period has expired.
GXSError	The platform is no longer stable. Commands may not work as expected. Restart the OTA Manager platform.
ListenerNotFoundException	Listener not found.
NoMatchingReceiverException	There is no matching receiver.
NotProcessingStateException	Not in processing state.
PlatformException	See contextual message.

invocationregistry Package

The **com.gemplus.gxs.internal.basesystem.invocationregistry** package throws the exceptions shown in the following table:

Table 76 - Invocation Registry Package Errors

Java Class	Definition
InvocationRegistryException	General exception of invocation registry.
MalformedSQLRequestException	Invalid SQL request.
DuplicatedInvocationIdentifierException	Identifier is duplicated
UnknownInvocationIdentifierException	Unable to update the given invocation: unknown identifier.

snmp Package

The **com.gemplus.gxs.internal.basesystem.snmp** package throws the exceptions shown in the following table:

Table 77 - SNMP Package Errors

Java Class	Definition
ParseMIBException	Problem when parsing XML file that contains the MIB description.
SNMPException	SNMP problem
InvalidPrimitiveException	Invalid SNMP primitive or name.
SNMPException	The OID cannot be truncated.

container Package

The **com.gemplus.gxs.internal.util.container** package throws the exceptions shown in the following table:

Table 78 - Container Package Errors

Java Class	Definition
IllegalServiceDomainException	Illegal service domain.
IllegalServiceTypeException	Illegal service type.
MissingPropertyException	A property is missing.
NoMatchingServiceClassException	No interface is defined for this service.
ServiceAccessDeniedException	You do not have the required user profile authorizations to access this service.
ServiceBreakdownException	The product or component used is down.

pckExceptions Package

The **com.gemplus.gemcast.common.pckExceptions** package throws the exceptions shown in the following table:

Table 79 - pckExceptions Package Errors

Java Class	Meaning
ServerException	General server exception.

pckTimeManagement Package

The **com.gemplus.gemcast.common.pckTimeManagement** package throws the exception shown in the following table:

Table 80 - pckTimeManagement Package Errors

Java Class	Meaning
GetGMTTimeException	Unable to get the GMT time.

sequencer Package

The **com.gemplus.gxs.internal.util.sequencer** package throws the exceptions shown in the following table:

Table 81 - Sequencer Package Errors

Java Class ²	Meaning
ClosedException	Closed exception (sequencer).
InvalidStateException	Invalid state (sequencer).
SequencerException	General sequencer exception.
UnknownTaskException	Task unknown (sequencer).

threadpool Package

The **com.gemplus.components.threadpool** package throws the exceptions shown in the following table:

Table 82 - Threadpool Package Errors

Java Class ²	Meaning
ClosedException	Closed exception (threadpool).
ThreadPoolNestedException	Thread pool nested.
ObjectPoolException	Problem with object pool.

toplink Package

The **Toplink** package throws the exceptions shown in the following table:

Table 83 - Toplink package errors

Java Class	Meaning
Public.Remote.CommunicationException	Remote communications problem with database.
Public.Exceptions.DatabaseException	Database problem.

java.lang Package

The **java.lang** package throws the exceptions shown in the following table:

Table 84 - java.lang Errors

Java Class	Meaning
Exception	General exception.
RuntimeException	Runtime exception.
IllegalArgumentException	The argument passed to the method is invalid.
NullPointerException	You are trying to use a null object.
ArrayIndexOutOfBoundsException	Check array length against number of parameters passed.
ClassCastException	You may encounter archive incompatibility between server and client.
IllegalStateException	A component is not in the expected state.
NoSuchMethodError	Check the build compatibility.
NumberFormatException	Error in number format (may be null or invalid String).

java.io Package

The **java.io** package throws the exceptions shown in the following table:

Table 85 - java.io Package Errors

Java Class	Meaning
IOException	Problem with stream.
FileNotFoundException	The file or directory you want to use does not exist.

java.sql Package

The **java.sql** package throws the exception shown in the following table:

Table 86 - java.sql Package Errors

Java Class	Meaning
SQLException	Invalid SQL command or SQL command does not match database settings.

java.net Package

The **java.net** package throws the exceptions shown in the following table:

Table 87 - java.net Package Errors

Java Class	Meaning
UnknownHostException	The remote host does not exist or is down.
ConnectException	Impossible to connect to the remote host; remote host is not listening on the expected port.
BindException	Impossible to connect to the specified port; port may be busy.

CCI Error Codes

The CCI throws the following types of exceptions:

- CciException
- MediationException
- BeanException
- ExternalException

The error codes for these exceptions are documented in the following tables.

CciException Error Codes

“Figure 88” lists CciException error codes:

Table 88 - CCiException Error Codes

Error Code	Meaning
0	Unknown error
1	Invalid argument
2	Invalid state
3	Too many requests, do a new filter
4	No request found, do a new filter
5	Invocation not found. Refresh
6	Invalid state or invocation not found
7	I/O error
10	The campaign is not found
11	The format of the imported file is incorrect
12	File type not defined in ‘dataParser.properties’ resource
13	Invalid request(s)
14	Request submitted without guarantee of execution
15	Invalid campaign(s)
20	Multi-card destinations with this service is deprecated. Use a campaign to do this
100	Bad record length

Table 88 - CCiException Error Codes (continued)

Error Code	Meaning
101	Class file interpreter not found
102	Class file interpreter instantiation error
103	Malformed card file content
104	File identifier not found for file interpreter
105	Access error
200	Product name is not specified
201	Invalid product
210	Target file name is not specified
211	Target directory path is not specified
212	Source file name is not specified
213	Source directory path is not specified
300	An MSISDN already exists for the SIM card
310	The SIM card is already active
320	Invalid SIM card
330	The MSISDN already exists in the Card database
340	Subscriber with web access is created
341	Subscriber without web access is created
350	An account has already been created with the specified identifier
351	A profile has already been created with the specified identifier
400	XML parsing error
401	Import file not found
402	Service class not found or instantiation impossible
403	User not authorized to perform this action
404	No services in connected user profile
405	No services in associated card profile
406	Request destination not associated to any profiles
407	Request destination not found
408	Error during request destination management
500	The extension of the selected file is invalid. Valid extensions are "cab", "hex", "s19", "mep" and "props"
505	The format of the selected file is not correct
510	The specified file cannot be found, check the path
600	The campaign cannot be found
605	The scenario cannot be found
610	The campaign file is invalid.

Table 88 - CCiException Error Codes (continued)

Error Code	Meaning
620	The campaign time frames are not valid
630	The search query must conform to the following pattern: searchCriteria [logicalOperator searchCriteria] searchCriteria ::= criteria operator 'value' criteria ::= serialNumber cardProfile cardState msisdnNumber imsiNumber operator ::= < >= <= = logicalOperator ::= and or
640	The label is already used by another scenario
650	The label is already used by another campaign
660	The scheduling information is invalid (beginning and ending date)
670	An implementation already exists for this service, use the provided page
680	No access right is defined for this service
700	The card is already defined in the database
701	The card you want to use or modify does not exist in the OTA Manager platform. In the campaign destination page, the card import from an XML file failed
702	The size of the imported data does not match with the file data size
703	The file you want to import does not exist or is empty
800	Failed to retrieve the SIM card set from the XML file. Check XML document. In the campaign scheduling page; retry delay value must be greater or equal to 60.
810	The retry delay must be an integer between 60 and 604800 (7 days)
811	This is not an integer value
812	The priority level must be an integer between 1 and 100
813	The Maximum number of retries must be an integer greater than or equal to 0
814	The Validity Period/service must be an integer between 60 and 86400 (24 hours)
900	The group is already defined in the database

MediationException Error Codes

"Figure 89" lists MediationException error codes:

Table 89 - MediationException Error Codes

Error Code	Meaning
20000	Platform access denied.
20001	Platform account expired.
20002	Operational channel not initialized.
20003	Management channel not initialized.
20004	Database connection not initialized.
20005	Operational message not sent.
20006	Cannot execute the request.
20007	Card Manager channel not initialized.

Table 89 - MediationException Error Codes (continued)

Error Code	Meaning
20008	Card Manager name not found in list of properties.
20009	Logon failed: Illegal authentication mode.
20010	Platform connection not initialized.
20011	Account manager not initialized.
20012	Profile manager not initialized.
20013	SNMP manager not initialized.
20014	Subscriber account manager not initialized.
20015	System configuration manager not initialized.
20016	Product manager not initialized.
20017	Billing manager not initialized.
20018	Logging manager not initialized.
20019	Audit Trail manager not initialized.
20020	Invocation manager not initialized.
20021	System time manager not initialized.
20022	Database error.
20023	Campaign facility not initialized.
20024	Campaign manager not initialized.
20025	Campaign scenario manager not initialized.
20030	The platform is probably down; contact the administrator.

BeanException Error Codes

“Figure 90” lists BeanException error codes:

Table 90 - BeanException Error Codes

Error Code	Meaning
10010	A mandatory field is not specified.
10020	A numeric field is not correct.
10030	A date is not in the format dd/mm/yyyy hh:mm:ss
10031	A time is not in the format hh:mm:ss
10040	Unable to open the file.
10050	Cannot find the object with the specified identifier.
10060	A hexadecimal field is not correct.
10080	A numeric field is out of range.
10090	In files structure, parent is not found, or is a final element.

ExternalExceptions Error Codes

“Figure 91” lists ExternalExceptions error codes:

Table 91 - ExternalExceptions Error Codes

Error Code	Meaning
30000	External error (from platform)

Javascript Message Labels

“Figure 92” lists JavaScript message labels

Table 92 - Javascript Message Labels

Java Class	Meaning
cci.javascript.error.requiredField	A mandatory field is not specified
cci.javascript.error.numericField	A number field is not correct
cci.javascript.error.hexadecimalField	A hexadecimal-coded field is not correct
cci.javascript.error.passwordField	The password is not correct (6 characters with at least 2 numbers)
cci.javascript.error.splIntegerField	A strictly positive integer field is not correct

Card Manager (RCA) Error Codes

“Figure 93” lists Card Manager (RCA) error codes:

Table 93 - Card Manager (RCA) Error Codes

Error Code	Meaning
RCA-1	Card is not active. You should activate the card
RCA-2	Profile is not active. You should activate the profile
RCA-3	Object already defined. You should specify another identifier
RCA-4	Object not found
RCA-5	Wrong service state. Take care of service state before performing this action
RCA-6	Bad service declaration. Check service declaration
RCA-8	Formatting library error. Consult the formatting library error code
RCA-10	Access forbidden. Verify user access
RCA-12	List is empty
RCA-13	Bad service definition. Check service definition
RCA-14	Unhandled exception. Internal error
RCA-15	Filter not found
RCA-16	Request deleted by user

Table 93 - Card Manager (RCA) Error Codes (continued)

Error Code	Meaning
RCA-17	Request not found
RCA-18	Service not found. Check service declaration
RCA-19	Channel not found. Verify protocol submitted
RCA-20	Connection timeout. Retry later
RCA-21	No channel for this protocol
RCA-22	Sending router error. Internal error
RCA-23	Unknown protocol. Verify protocol name
RCA-24	Component not found. Internal error
RCA-25	Incompatible data type. Internal error
RCA-26	Request execution partially successful
RCA-27	Request execution failed
RCA-28	Request not processed, product shutdown
RCA-29	Request sent but product shutdown
RCA-30	Request sent but CMG connection lost
RCA-31	The formatting service raised an error. Please check the error code of the embedded library
RCA-32	Access, update or commit on a Card Security does not work. Verify the ICCID validity
RCA-33	The requested formatting type exists, but is not active. Verify its activation in the management interface
RCA-34	The requested formatting type is not handled. Verify the existing formatting types in the management interface
RCA-35	Creation, update or deletion on a transport key does not work. Check the transport key index and hexadecimal value
RCA-36	Product is not connected
RCA-37	Com problem
RCA-38	Request is expired
RCA-39	Card not found
RCA-701	Driver class could not be found
RCA-702	Incorrect BaseSystem parameter value
RCA-703	Incorrect parameter value
RCA-706	SMSC reference could not be found
RCA-709	Channel Driver SMS error
RCA-711	Channel Monitor SMS error
RCA-712	Channel Monitor WIR error
RCA-718	Impossible to connect
RCA-719	No Login response. Connection failed

Table 93 - Card Manager (RCA) Error Codes (continued)

Error Code	Meaning
RCA-720	Driver initialisation error
RCA-722	Impossible to find an available message ID
RCA-723	Invalid incoming message
RCA-724	SMSC is no longer alive
RCA-727	Driver Busy for Message
RCA-729	Failure sending Nokia channel driver. This error code is returned by the SMSC—refer to the SMSC documentation for further details
RCA-730	Failure sending SMPP channel driver. This error code is returned by the SMSC—refer to the SMSC documentation for further details
RCA-731	Failure sending CMG channel driver. This error code is returned by the SMSC—refer to the SMSC documentation for further details
RCA-732	Failure sending SEMA channel driver. This error code is returned by the SMSC—refer to the SMSC documentation for further details
RCA-733	Failure delivering Nokia channel driver. This error code is returned by the SMSC—refer to the SMSC documentation for further details
RCA-734	Failure delivering SMPP channel driver. This error code is returned by the SMSC—refer to the SMSC documentation for further details
RCA-735	Failure delivering CMG channel driver. This error code is returned by the SMSC—refer to the SMSC documentation for further details
RCA-736	Failure delivering SEMA channel driver. This error code is returned by the SMSC—refer to the SMSC documentation for further details
RCA-738	Channel driver already exists
RCA-739	Channel driver does not exist
RCA-740	Unexpected exception
RCA-1000	Invocation Manager unexpected error
RCA-1001	Invalid invocation
RCA-1002	Expired invocation
RCA-1003	Deleted invocation
RCA-1004	Invocation not completed
RCA-1005	Access denied for invocation submission
RCA-1200	Service declaration not active. Activate it
RCA-1201	Unknown service declaration. Declare service or check service name
RCA-1202	Unknown service implementation. Check for declared services in the profile
RCA-1203	Component not active. Activate It
RCA-1204	Missing mandatory parameter. Specify it
RCA-1205	Internal error
RCA-1206	Timeout occurred while formatting
RCA-1207	Bad parameter format. Check format

Table 93 - Card Manager (RCA) Error Codes (continued)

Error Code	Meaning
RCA-1208	Bad request format. Check format
RCA-1209	Some items of information are missing. Specify missing items
RCA-1210	Data consistency error. Check data consistency
RCA-1300	Card profile is not CAT-TP compatible. Check the card profile's capabilities list
RCA-1301	Service is not CAT-TP compatible
RCA-1302	Service is not CAT-TP compatible because is not 03.48 compliant. Check the TAR value
RCA-1303	CAT-TP connection lost during service execution
RCA-1304	No CAT-TP port defined for targeted application. Check the card profile's Java applet instance properties list
RCA-1305	Communication with remote entity failed
RCA-1306	Device is not BIP capable
RCA-1307	Error during call of plugged DeviceCapabilitiesHandler
RCA-1310	The PoR requested for CAT-TP connection establishment not received
RCA-1311	CAT-TP connection not established and additional data content if present in the PoR
RCA-1312	Service expired before a CAT-TP connection could be established
RCA-1320	CAT-TP connection closed before PoR received
RCA-1321	Service validity period reached before service execution completed
RCA-1325	PUSH SMS: data too long and cannot be sent using a single SMS. Check the value of the SMS_PUSH_ALPHA_ID product parameter (see "Card Manager (RCA) Parameters" on page 173)
RCA-1401	Subscriber status handler adaptor exception
RCA-1402	Invocation is not reachable for this subscriber
RCA-1410	Unable to instantiate application. This service cannot instantiate an application with the "Security Domain" privilege
RCA-1411	Unable to manage application (Java Applet instance). This service cannot manage an application with the "Security Domain" privilege
RCA-1412	Unable to install security domain. Card content indicates that this package does not exist
RCA-1413	Unable to install security domain. Card content indicates that a security domain with the same AID already exists
RCA-1420	Service execution failed because of an invalid PoR
RCA-1421	Unable to manage security domain. This service cannot manage an application that is not a security domain
RCA-1430	Unable to change security domain status. Card content indicates that the security domain does not exist
RCA-1431	Unable to change security domain status. Card content indicates that the security domain's status is invalid
RCA-1440	Unable to delete security domain. Card content indicates that the security domain does not exist

Table 93 - Card Manager (RCA) Error Codes (continued)

Error Code	Meaning
RCA-1445	Unable to manage (delete, add, or change) key. Card content indicates that the security domain does not exist
RCA-1446	Unable to manage (delete, add, or change) key. Card content indicates that the target is not a security domain
RCA-1450	Unable to extradite entity (Java package or Java applet instance). Card content indicates that the entity is missing from the card
RCA-1451	Unable to extradite entity (Java package or Java applet instance). Card content indicates that the entity already belongs to the specified security domain
RCA-1500	Token generation failed. No security domain with the "Token Verification" privilege is defined for the card
RCA-1501	Token generation failed. Security domain with the "Token Verification" privilege for the targeted key set does not exist
RCA-1502	Token generation failed. Missing mandatory information in command parameters
RCA-1503	Token generation failed. Invalid format of the command parameters field
RCA-1504	Token generation failed. Targeted security domain does not have the "Delegated Management" privilege
RCA-1510	Receipt notification failed. No security domain with the "Receipt Generation" privilege is defined for the card
RCA-1511	Receipt notification failed. Security domain with the "Receipt Generation" privilege for the targeted key set does not exist
RCA-1512	Receipt notification failed. Missing mandatory information in command parameters
RCA-1513	Receipt notification failed. Invalid format of Data Parameters Confirmation field
RCA-1514	Receipt notification failed. Targeted security domain does not have the "Delegated Management" privilege
RCA-1515	Receipt notification failed. Unable to retrieve the related token request

Campaign Manager Error Codes

Errors Originating From the Product Connector

The following errors are generated by the **com.gemplus.components.gemcee.abstractsee** package, indicating an error with the product connector:

Table 94 - Product Error Messages

Java Class	Meaning
IllegalAbstractSEESTateException	This exception is sent when there is a problem in the underlying product that prevents any operations being executed.
NoSuchAbstractSEEException	This exception is returned by the Campaign Manager product connector when it cannot find the specified product reference

Errors Originating From Campaigns

The following errors are generated by the **com.gemplus.components.gemcee.campaign** package, indicating an error with the campaign execution:

Table 95 - Campaign Error Messages

Java Class	Meaning
CampaignAlreadyExistsException	This exception is raised when you try to give a new campaign a name that is already used by an existing campaign
IllegalCampaignStateException	This exception is raised when you try to perform an action that is not allowed at current state of the campaign
NoMatchingTargetException	This exception is raised when Campaign Manager tries to execute an action with a specified criteria that does not match any of the existing campaign targets.
NoSuchCampaignException	This exception is raised when a user tries to perform an action on a campaign and Campaign Manager cannot find the specified campaign.
NoSuchScenarioElementException	This exception is raised when Campaign Manager tries to execute a service that doesn't exist in the scenario.
NoSuchTargetException	This exception is raised when the user tries to select a target that does not exist. Someone has removed the relevant line from the database.
TimeFrameOverlapException	This exception is raised when you try to define a time frame that overlaps an existing time frame.
TooManyTimeFramesException	This exception is raised when you try to add a new time frame when the maximum number of time frames for the same campaign has already been reached.
UnreadableFileNotFoundException	This exception is raised when Campaign Manager cannot read a target list file.
UnwritableFileNotFoundException	This exception is raised when an "export" file (.target,.XML) cannot be written to.

Errors Originating From Campaign Scenarios

The following errors are generated by the **com.gemplus.components.gemcee.campaign.scenario** package, indicating an error with the campaign scenario execution:

Table 96 - Campaign Scenario Error Messages

Java Class	Meaning
IllegalScenarioStateException	This exception is raised when you try to modify a scenario that is currently being used by at least one campaign, even if the campaign is TERMINATED.
NoSuchScenarioException	This exception is raised when you try to assign a scenario to a campaign and Campaign Manager cannot find the campaign.
ScenarioAlreadyExistsException	This exception is raised when you try to create a new scenario with a name that is already being used by an existing scenario.

General Errors

The following are general errors generated by the **com.gemplus.components.gemsee.engine**:

Table 97 - General Error Messages

Java Class	Meaning
AccessDeniedException	This exception is raised when a user tries to perform an action without having the necessary access rights.
DatabaseException	This exception is raised when there is a problem with the database.
PersistenceException	This exception is raised when there is a problem with the hardware.

XCT Error Codes

Two types of error codes are managed in XCT:

- Errors specified by the XCT module: this is the list of “error codes” that follow in this chapter.
- “Error details” specified by external software (for example, the SMSC, RCA or Oracle Database). These errors are not listed in this manual because they depend on the external software (Logica SMSC errors, BMD SS7 router errors, CMG SMSC errors, RCA errors, and Oracle Database errors all differ).

In one case, for two different errors detected by OTA Manager V5.1, a single error code has been used with two different error details. This case is an external provisioning operation occurring during a XCT campaign: error code “1012” is used if detected during pre-processing phase, “2009” if detected during sending, or “3010” if detected during post-processing.

The four error details are:

- 1 If the MSISDN has been set to null for a given card (this is the case if a subscriber changes his card: the MSISDN associated to the “old” card is first to “null”, then the MSISDN is associated to the “new” card)
- 2 If the MSISDN has been changed for a given card.
- 3 If the card has been deleted.
- 4 If the card’s state has been changed to “marked for deletion”.

XCT Global Errors

The following tables lists global XCT errors:

Table 98 - Global XCT Errors

Error Code	Meaning
1	Database is full
2	Unable to log on to database
3	Invalid MSISDN
4	Number of retries is not valid
5	Invocation identifier is not valid
6	Unable to set Oracle Debug mode
7	Unable to unset Oracle Debug mode
8	Invalid SMSC reference
9	Invalid Message identifier
10	Invalid Status identifier
11	Invalid Campaign identifier
12	No Campaign running
13	No scenario found
14	Invalid module for timeslot management

Table 98 - Global XCT Errors (continued)

Error Code	Meaning
15	Invalid day for timeslot management
16	Invalid interval for timeslot management
17	Unable to get CORBA proxy

XCT Pre-Processing Errors

Table 99 - XCT Pre-Processing Errors

Error Code	Meaning
1000	General error
1001	Card Manager (RCA) formatting error
1002	Service execution error. The following values are possible: 5 : Wrong service state 8 : Formatting library error 13 : Bad service definition 18 : Service not found 23 : Unknown protocol 25 : Incompatible data type. Internal error 31 : The formatting service raised an error. Check the error code of the embedded library 33 : The requested formatting type exists, but is not active 34 : The requested formatting type is not handled 37 : Communication Problem 1200 : Service declaration not active 1206 : Timeout occurred while formatting
1003	OTA Manager error
1004	SQL error. The error code returned from Oracle. Refer to the Oracle documentation for details.
1005	Unable to get Card Manager (RCA) pre-processing service. This message only occurs in the log file.
1006	Invalid Card Manager (RCA) arguments
1007	Invalid campaign arguments
1008	Unable to lock card
1009	Unable to insert SMS. The Oracle error code, or 1 for any other problem
1010	Unable to insert synchronisation counter. The Oracle error code, or 1 for any other problem
1011	Unable to insert card content. The Oracle error code, or 1 for any other problem
1012	An external card operation occurred. The following values are possible: 1 : Card updated : desactivated - No MSISDN 2 : Card updated : replace new MSISDN 3 : Card deleted 4 : Card maked for deletion
1013	Invocation could not be treated during the scheduled campaign time.

XCT Sending Errors

Table 100 - XCT Sending Errors

Error Code	Meaning
2000	Invalid SMS body: NULL in database. This message only occurs in the log file
2001	Unable to decrease flow control value. This message only occurs in the log file
2002	Invalid SMSC type. This message only occurs in the log file
2003	SMSC was disconnected. Null, or the SMS packet ID if an incorrect WINDOW_SIZE parameter value was specified
2004	Submit Response failed. The SMS submit code
2005	Delivery Status or Query Response failed. The SMS delivery code
2006	Delivery Status not received but Inquiry activated
2007	Delivery Status interpreted as expired. The SMS delivery code
2008	No PoR received
2009	An external card operation occurred. The following values are possible: 1 : Card updated : desactivated - No MSISDN 2 : Card updated : replace new MSISDN 3 : Card deleted 4 : Card maked for deletion
2010	Invocation contains no SMS to send
2011	SMS is incorrectly formatted. The SMS packet identifier

XCT Post-Processing Errors

Table 101 - XCT Post-Processing Errors

Error Code	Meaning
3000	General error
3001	Unable to retrieve profile data used during post-processing. The OTA Manager profile identifier.
3002	Unable to retrieve PoR during post-processing
3003	Unable to unformat PoR
3004	Unable to get Card Manager (RCA) post-processing service
3005	Invalid arguments
3006	SQL error. The Oracle error code is displayed. Refer to the Oracle documentation for details.
3007	Status Code in PoR is failed. The status code is displayed.
3008	Additional data in PoR are failed
3009	Unable to insert additional data. The Oracle error code is displayed. Refer to the Oracle documentation for details.

Table 101 - XCT Post-Processing Errors

Error Code	Meaning
3010	An external card operation occurred. The following values are possible: 1 : Card updated : desactivated - No MSISDN 2 : Card updated : replace new MSISDN 3 : Card deleted 4 : Card maked for deletion
3011	Unable to update the synchronization counter. The OTA Manager profile identifier.
3012	SMS sending is not completed
3013	Profile data used during the post-processing phase is not coherent
3014	One or more service executions of a multiple services campaign scenario failed.
3015	Status word in PoR is failed
3016	Unable to extract status code in PoR
3017	Unable to update card content

Troubleshooting

Startup Problems

If OTA Manager V5.1 Doesn't Start

Check 1

Check that Oracle Database is started.

How:

To see the Oracle Database processes running, use the command:

```
ps -eaf | grep -i ora
```

Check 2

Check that the Oracle Database Listener is started.

How:

To see the Oracle Database Listener process running, use the command:

```
ps -eaf | grep -i tnslsnr
```

Check 3

Check there is no conflict with the VisiBroker port.

How:

To see available ports, use the command:

```
netstat -a
```

Impossible to Access the Management Interface

Check 1

Check if the Web server is started.

To see if the Tomcat process is running, use the command:

```
/usr/ucb/ps -auxww | grep -i tomcat ****use tomcat -state****
```

Check 2

Check if there is an access problem due to a firewall.

Problems Sending SMS Messages

The following description assumes that the service request's parameters have been filled in, the request submitted, and the "Request Submitted" message is displayed in the management interface. Monitoring of the request is performed using the Request Monitoring window shown in "Figure 70" (choose **Request > Monitoring** from the left-hand menu):

Figure 70 - The Request Monitoring Window

The screenshot shows the Request Monitoring window with the following sections:

- Criteria setting**: Contains filtering criteria for On-going (Created, In process) and Processed (Succeeded, Failed, Deleted) requests.
- Filtering criteria**: Shows the same categories as the Criteria setting section.
- Sorting criteria**: Allows sorting by Available fields (Destination, Submit Date, Service, User) in Ascending or Descending order.
- Query creation**: Includes a checkbox for "Only own requests" and a query definition section. The query definition section contains a dropdown for Destination, an equals sign, and a dropdown for another field. Buttons for Add, Replace, Modify, and Remove are available for the query definition.
- Results per page**: Set to 35, with View... and Clear buttons.

The request is not present on the monitoring page

Check 1

Verify that the request **Filtering criteria** of the Request Monitoring window allows the request to be selected.

The simplest method is not to specify any criteria; all requests are displayed in this case.

Check 2

If the request has been submitted by another user, check that you are authorized to view requests submitted by other users; only administrators have the right to view all the requests of the system. Customer care agents and subscribers are only allowed to view their own requests, that is, requests they originally submitted.

Check 3

Requests selected on the Request Monitoring window are displayed by blocks (the size of each block is determined by the value of the **Results per page** field at the bottom of the window).

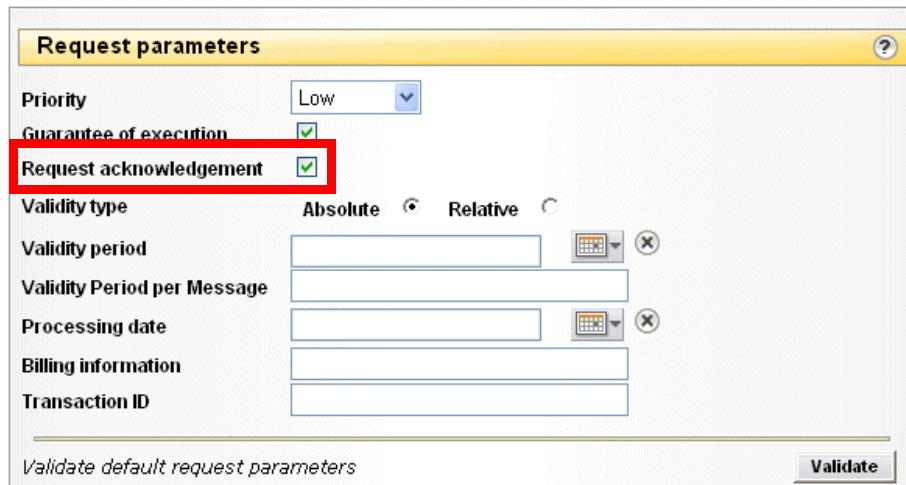
View all blocks of requests: the request is perhaps not shown on the first block. Click the **Next block** button in order to view the next block of requests.

Check 4

Check the log file or the billing file to determine whether the request has terminated. If it has, this indicates that the request has been executed correctly. By default, requests that complete successfully are automatically removed from the platform, and therefore are no longer accessible from the Request Monitoring window.

If you want the request to still be available on the Request Monitoring window despite successful completion, select the **Request acknowledgement** check box on the Request Parameters window when submitting the request:

Figure 71 - The Request Parameters Window



The request is present on the monitoring page, but its status is “FAILED”

Check 1

Request execution has failed. View the explanation of the error by accessing detailed information on the request:

- 1 Click the  icon
- 2 Select the targeted card in the **Destination** list.

The error message is displayed in the **Submission Result** field.

Processing errors can occur for the following reasons:

- The service's parameters are incorrect
- The OTA service did not execute correctly
- The targeted card is not registered with the OTA Manager platform
- The security keys of the target card are not correctly provisioned and could not be found by OTA Manager when attempting to format SMS messages.

- The security settings specified for the service are not compliant with the security information specified for the card (for example, the specified security level does not exist).

Check 2

Check the log file to find out at which step request execution failed and to obtain more information on the error.

The request is present on the monitoring page, but its status remains “CREATED” for a long time

Check 1

The platform may be temporarily overloaded, meaning that the request cannot be processed for the moment.

To check this, go to the Platform Supervision window.

Note: Only administrators are authorized to view this window. Ask an administrator to help if you do not have sufficient access rights.

Check whether any **SNMP Trap** bullets are red, indicating that a problem occurred and an SNMP trap was sent to the SNMP Manager. Check in particular (in the **Product List** area of the window) for traps relating to the Invocation Registry database and the Card Manager.

Also check the counters in the **Invocation manager** area of the window: if the **Accepted** counter **Request In** column is growing whereas the number of requests in the **Output** columns is not increasing, this might indicate that the platform is overloaded.

In case of such problems, contact your system administrator for a more detailed analysis of the SNMP counters to establish a better indication of the status of the platform.

Check 2

Check whether there is activity (corresponding to other requests) in the **Channel Driver** area of the window. To do this, click on the  icon of the **Channel Driver** list.

First check that the channel driver is connected. If it is and there is still no output activity, this might indicate a problem within the channel driver. Try to deactivate and reactivate the driver (click **Deactivate Driver** then **Activate Driver** on the Driver Management window).

If there is still no output, check the status of the SMSC.

The request is present on the monitoring page, but its status remains “IN PROCESS” for a long time

This first step in this situation is obtain the current service execution step of the request. To do this, on the Request Monitoring window, click on the  icon of the request. The detailed status of the execution of the request is displayed in the **Destination** table.

IN PROCESS Status

If the status is IN PROCESS, the request is currently being processed by the platform (retrieving necessary information from databases, formatting SMS messages, and so on).

Check 1

Check whether there is another request currently in execution for the same card. If so, the current request will remain on hold until the previous request has completed, as there can only be one request sending messages to a card at any one time.

Remember that if you are customer care agent, you are only able to view the requests you have submitted. Ask an administrator to check all the requests of the platform.

In order for your request to continue execution, wait for the previous request to complete, the validity period to expire, or try to delete the previous request by clicking the **Delete** button on the Request Monitoring window.

Check 2

If there is no other request for the same card, check the log file for information to help you understand the problem.

SENDING Status

If the status is SENDING, the SMS messages of the request are in the process of being sent.

Check 1

First, check that the number of SMS messages displayed in the **Destination** table (**SMS Number** field) is increasing over time. If so, then the sending phase is executing correctly, but seems to be lasting a long time. Check the SMSC load and response time (by either inspecting the platform log file, checking the time between the SMS message being sent to the SMSC and the SMSC acknowledgement being received by the platform, or by inspecting the SMSC logs and statistics directly).

Check 2

If the number of SMS messages is not increasing, check for activity (corresponding to other requests) in the SMS channel driver. To do this, click on the  icon of the **Channel Driver** list of the Platform Supervision window.

If there is some activity, the SMS messages will probably be sent shortly.

Check 3

If there is no channel driver activity, check that the SMSC driver is active.

Display the Driver Management window, as described in “Activating and Deactivating the Audit Trail” on page 32.

If the SMSC driver is inactive (, activate it by clicking on the **Activate Driver** button of the Driver Management window, then wait for the driver to connect automatically.

Check 4

If the SMSC driver is active but not connected (, try to force it to reconnect the SMSC. To do so, deactivate and reactivate the driver (click **Deactivate Driver** then **Activate Driver** on the Driver Management window).

Check 5

If the driver is still not connected, search the log file to discover why the driver connection has failed.

Check 6

Check the log file to locate any other problems.

WAIT POR status

If this status is WAIT POR, this means that the SMS messages have been sent to the card, and the platform is waiting for the proof of receipt (PoR) to arrive from the card.

Check 1

If you have access to the mobile equipment, check that it has network coverage, appears to be processing SMS messages correctly, and seems generally to be working correctly.

Check 2

Check whether there is MO activity (corresponding to the MO/PoR of other requests) on the SMS channel driver. To do so, click on the  icon of the **Channel Driver** list of the Platform Supervision window.

Check 3

Check that there is at least one channel driver active and connected, as described previously.

Any Other Status

Check 1

Check the log file to locate the other problem.

Problems with SMS Execution

On a native card, the SMS reaches the card but is not correctly executed

Check 1

Check that the originating address (TP-OA field) used by OTA Manager V5.1 is the same as that stored in the target card.

Check 2

Check that the synchronization counter value sent by OTA Manager V5.1 is not smaller than the one in the card.

Check 3

Check that the OTA key has been correctly provisioned in the OTA platform.

Check 4

Check that the algorithm for the OTA key used by OTA Manager V5.1 is the one configured in the card.

On a Java card, the SMS reaches the card but is not correctly executed on the card

Check 1

Check that the Minimum Security Level (MSL) used to send the SMS is compatible with the card.

Check 2

Check that the synchronization counter value sent by OTA Manager V5.1 is not lower than the value stored in the card (if used).

Check 3

Check that the OTA security keys have been correctly provisioned in the OTA platform.

Check 4

Check that the algorithm for the OTA security keys used by OTA Manager V5.1 is the one configured in the card.

Check 5

Check that the key set number used to target the GSM, GOP or USIM interpreter applets is the one configured in the card.

On a non-Gemalto card, SMS messages reach the card, but only the first one is correctly executed

Check 1

Check with the provider of the card that the card supports SMS concatenation.

SMS is not delivered

The following table is an extract from GSM 03.40 norm (release 7.5). It is the list of errors that can occur when the SMSC is trying to deliver a SMS-MT to a card. The error codes depend on SMSC implementation. Please refer to SMSC or protocol documentation.

Each error is classified as either temporary or permanent. This classification gives an indication of whether or not it is likely that the handset will become reachable within a reasonable period, and so provides the recommended action to be taken by the SMSC, (retry the message or discard it).

Table 102 - GSM 03.40 Errors

Error Indication	Type	Description
Unknown subscriber	P	The PLMN rejects the short message TPDU because there is no IMSI or directory number allocated to the mobile subscriber in the HLR (see GSM 09.02).
Teleservice not provisioned	P	The PLMN rejects the short message TPDU because the recipient MS has no SMS subscription (see GSM 09.02).
Call barred	T	The PLMN rejects the short message TPDU due to barring of the MS (see GSM 09.02, description of the Barring supplementary service, GSM 02.04 and 03.11), description of Call barred due to Unauthorised Message Originator, GSM 09.02, and description of Operator Determined Barring, GSM 02.41 and 03.15).
Facility not supported	T	The VPLMN rejects the short message TPDU due to no provision of the SMS in the VPLMN (see GSM 09.02).

Table 102 - GSM 03.40 Errors (continued)

Error Indication	Type	Description
Absent subscriber	T	<p>The PLMN rejects the short message TPDU because</p> <ul style="list-style-type: none"> ■ There was no paging response via the SGSN, MSC or both, (see GSM 04.08 & GMS 09.02) ■ The IMSI GPRS or both records are marked detached (see GSM 09.02), ■ The MS is subject to roaming restrictions (see "Roaming not allowed", GSM 09.02). ■ Deregistered in the HLR. The HLR does not have an MSC, SGSN or both numbers stored for the target MS, (see GSM 09.02) ■ Unidentified subscriber (see GSM 09.02) ■ MS purged, (see GMS 09.02) <p>(The reasons for absence are assigned integer values in table 03.40/1a. The appropriate integer value is sent with the absent subscriber error indication as defined in GSM 09.02)</p>
MS busy for MT SMS	T	<p>The PLMN rejects the short message TPDU because of congestion encountered at the visited MSC or the SGSN. Possible reasons include any of the following events in progress:</p> <ul style="list-style-type: none"> ■ Short message delivery from another SC ■ IMSI or GPRS detach ■ Location Update or Inter SGSN Routing Area Update ■ Paging ■ Emergency call ■ Call setup
SMS lower layer capabilities not provisioned	T	<p>The PLMN rejects the short message TPDU due to MS not being able to support the Short Message Service.</p> <p>The short message transfer attempt is rejected either due to information contained in the class mark, or the MSC not being able to establish connection at SAPI = 3 (see GSM 04.08 and GSM 09.02).</p>
Error in SMS	T	<p>The PLMN rejects the short message TPDU due to an error occurring within the MS at reception of a short message, e.g. lack of free memory capacity or protocol error.</p>
Illegal subscriber	P	<p>The PLMN rejects the short message TPDU because the MS failed authentication</p>
Illegal equipment	P	<p>The PLMN rejects the short message TPDU because the IMEI of the MS was black listed in the EIR</p>
System failure	T	<p>The PLMN rejects the short message TPDU due to network or protocol failure others than those listed above (see GSM 09.02)</p>
Memory capacity exceeded	T	<p>The MS rejects the short message since it has no memory capacity available to store the message</p>

Provisioning Problems

Loading XML files generates large numbers of errors

Check 1

Check that the XML files conform to the rules of the appropriate DTD (refer to "Appendix A - Batchloading XML Files").

Check 2

Check that the XML files loaded do not contain a control-M (^M) character at the end of each line (this may be placed there by certain text editors).

OTA keys loaded are erroneous

Check 1

Check whether the OTA keys are ciphered using a transport key by looking into the SIM card data files.

Check 2

If the OTA keys are not ciphered, no transport key is required.

Check with the provider of the cards that no error has been made when generating the OTA keys in the SIM card data files.

Check 3

If the OTA keys are ciphered, a transport key is required.

Check the transport key index value in the SIM card data file and check that the corresponding transport key has been correctly entered into OTA Manager V5.1.

Refer to the discussion of transport keys in “Transport Keys” on page 77.

Monitoring Problems

Log files are not created

Check 1

Check that the log files are activated in OTA Manager V5.1.

Refer to “Configuring the Core Framework” on page 11.

Check 2

If the log files are already in the Active state, check that there is no “file system full” message on the platform.

Check 3

If the log files are already in the Active state, check the access rights on the directory dedicated to the log files.

Audit trail files are not created

Check 1

Check that the Audit trail files are activated in OTA Manager V5.1.

Refer to “Configuring the Audit Trail” on page 14.

Check 2

If the Audit trail files are already in the Active state, check that there is no “file system full” message on the platform.

Check 3

If the Audit trail files are already in the Active state, check the access rights on the directory dedicated to the Audit trail files.

Billing files are not created

Check 1

Check that Billing ticket file generation is activated.

Refer to “Configuring Billing” on page 15.

Check 2

If the Billing files are already in the Active state, check that there is no “file system full” message on the platform.

Check 3

If the Billing files are already in the Active state, check the access rights on the directory dedicated to the Billing files.

Problems with the Wired Internet Reader (WIR)

Warning: Proof of Receipt (PoR) messages are not supported when using the WIR with SIMERA™ 2 cards.

If you encounter problems using the WIR (for example, the applet does not start correctly) check the following:

- **Multiple card readers**

Best solution: unplug unnecessary card readers—only one card reader should remain plugged in—restart the installer (it will just remove the entries in the registry but not uninstall the drivers), then reboot your computer.

It is also possible to manually check that only one card reader is defined in the registry. Go to:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography\Calais\Readers
and check that only one card reader is present.

- **Multiple JRE and browser**

The JRE version must be 1.4, 1.5 or 1.6. If not, restart the installer, download the suggested JRE and install it.

To ensure that the WIR applet can work in your browser, use the following link (official Sun page to test Java on your computer):

<http://www.java.com/en/download/help/testvm.xml>

If the applet does not start, follow the instructions on the Web page.

To check to ensure that the applet is working correctly, you must ensure that the plugin or add on inside the browser is consistent with the value of the JAVA_HOME environment variable then disable or uninstall unnecessary add ons:

- a) Choose Start > Run
- b) Type cmd
- c) Enter the command **java -version**

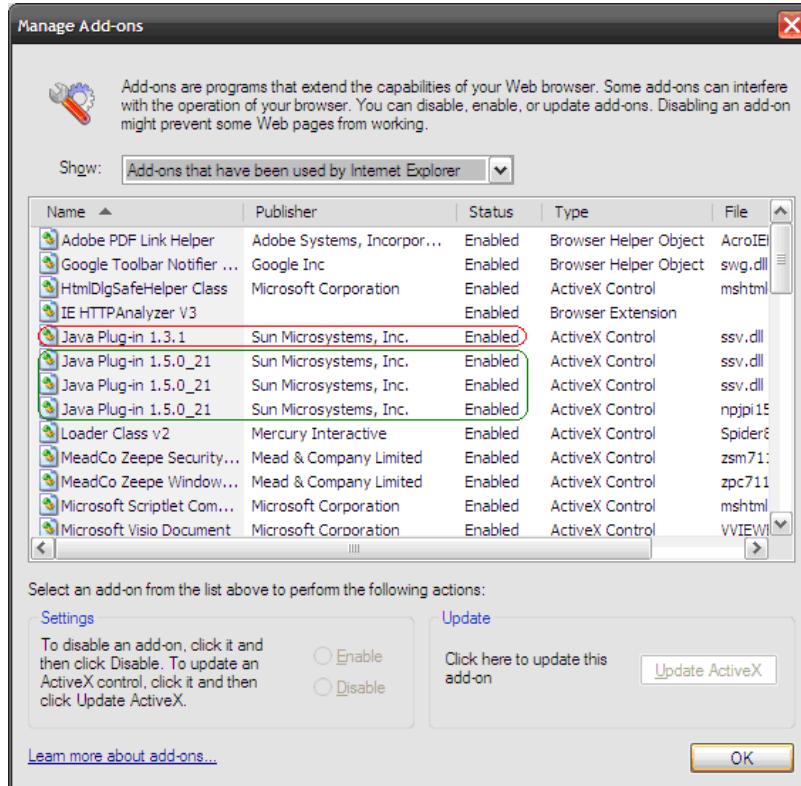
```
U:\>java -version
java version "1.5.0_21"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_21-b01)
Java HotSpot(TM) Client VM (build 1.5.0_21-b01, mixed mode, sharing)
```

The procedure depends on the browser you are using:

Internet Explorer

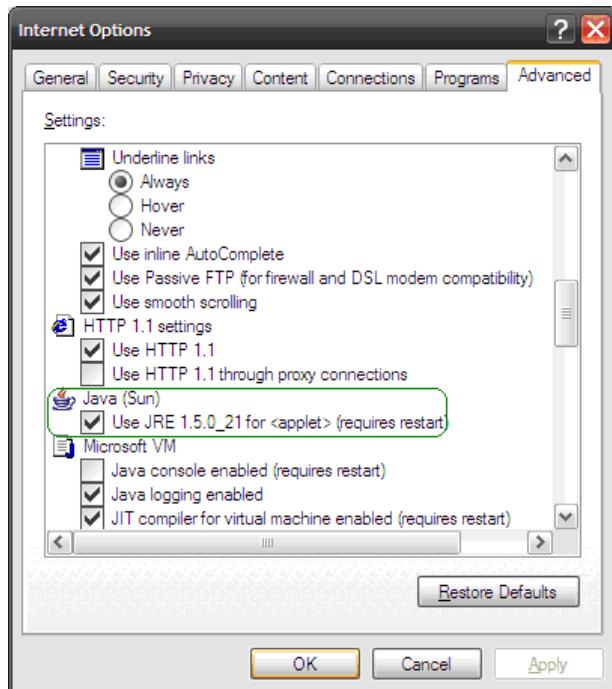
- a) Start Internet Explorer
- b) Choose **Tools > Internet Options > Programs > Manage Add-ons**
- c) Check that the Java Plug-in is consistent with the **java -version** output.

Figure 72 - Internet Explorer Add-Ons



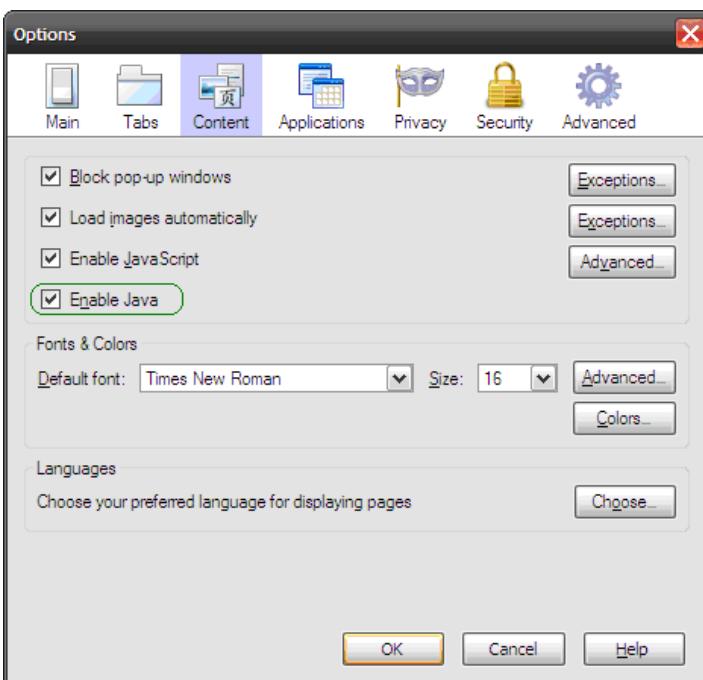
In this example, Java 1.3 must be disabled.

- d) Go to **Tools > Internet options > Advanced > Java (Sun)**
Check that **Use JRE XX for <applet>** is selected (checked):

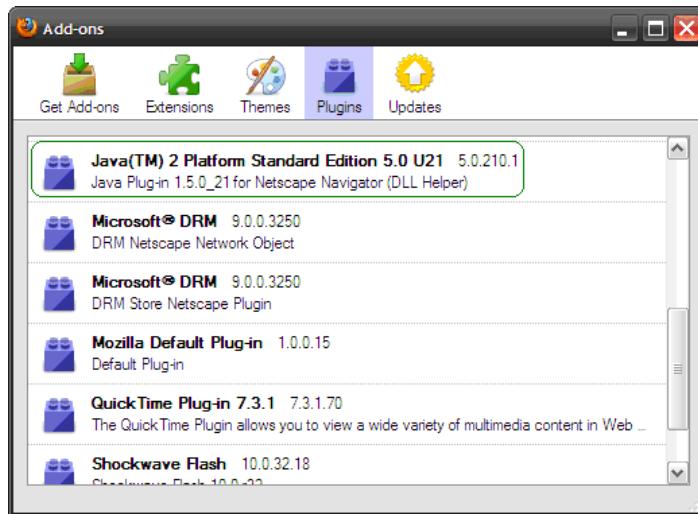
Figure 73 - Internet Options**Mozilla Firefox**

- Go to **Tools > Options > Content**

Check that Java is enabled:

Figure 74 - Mozilla Options

- Go to **Tools > Options > Main > Manage Add-ons > Plugins**
- Check that the correct Java version is activated:

Figure 75 - Mozilla Add-Ons

Disable other Java plug-ins.

Unique access the card reader

Make sure that all applications (Card Admin, SMSC simulator, and so on) that may be accessing the card reader are switched off.

If after using the WIR you cannot use other applications that connect to the card reader, check that the Java process is not still running. Normally, Java automatically stops by itself. However this may take some time. For this reason, either kill Java manually in the Task Manager or, if the process does not appear in the Task Manager, close your browser as the process can be embedded inside the browser process.

■ Libraries and certificates

Verify that certificates and libraries have been correctly copied to your JAVA_HOME directory:

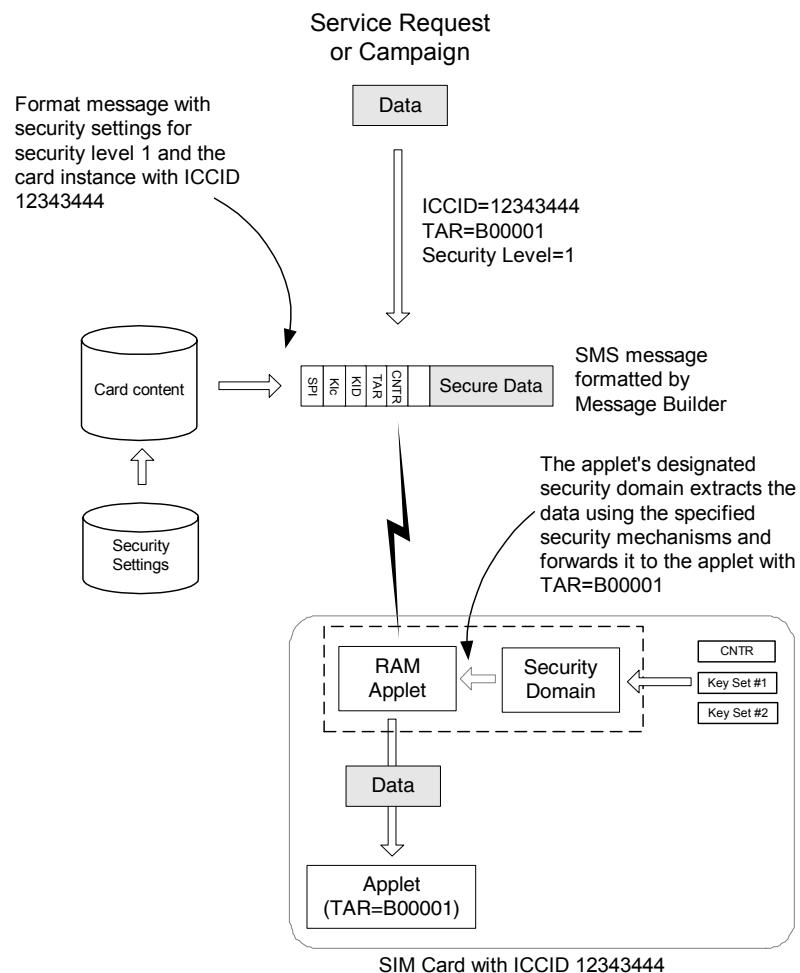
- Under JAVA_HOME/lib/ext:
gemplus-terminals-4.1.jar, OCFPCSC1.DLL, OCFPCSCM.DLL,
reference-services.jar, reference-terminals-windows.jar,
base-core.jar, base-opt.jar
- Under JAVA_HOME/lib/security:
cacerts, java.policy
- Under JAVA_HOME/lib:
opencard.properties

The “Security Domain” Security Model

Implementing Security

“Figure 76” shows an overview of OTA Manager’s Security Domain security model:

Figure 76 - The OTA Manager Security Model

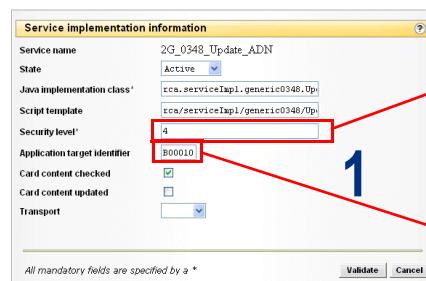


Overview

"Figure 77" illustrates how the Security Domain security model is implemented in the management interface.

Figure 77 - Overview of the Security Domain Security Model

- 3** A security level matching that specified in the service request is found in the entity properties



- 1** The platform searches in the card structure for an application with the TAR specified in the service request

- 2** From the TAR, the applet instance AID is found. The entity properties of the applet instance are inspected

The screenshot shows the "Card structure description (mNFC_v1.0)" dialog. It displays the "Java applet instance" section, which includes fields like "Instance AID" (highlighted with a red box and containing "A000000018434DFF33FFFF89000000"), "Status" (Selectable), "Size" (0), and "Configuration". A red arrow labeled "2" points from the "Instance AID" field to the corresponding step in the diagram below. Another red arrow labeled "3" points from the "Security level" field in the first screenshot to this dialog.

- 4** Inspecting the security domain associated with the applet instance, the KiC and KID values are used to extract the key values and synchronization counter values from the security settings.

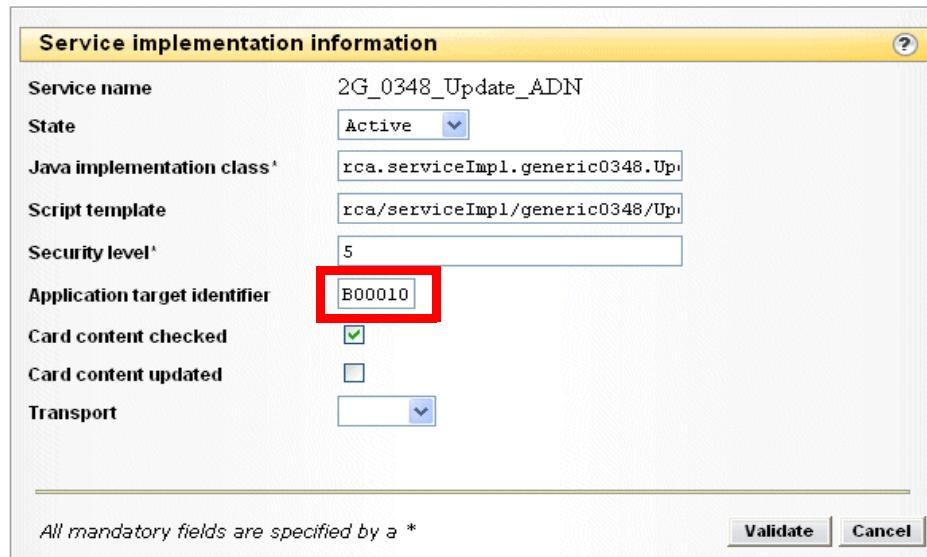
The screenshot shows the "03.48 security settings" dialog. It includes sections for "SPI command packet" (with "Cryptographic checksum (CC)" selected) and "KIC/KID settings" (showing KIC as "00" and KID as "10"). Red boxes highlight the "KIC" and "KID" fields. A red arrow labeled "4" points from the "KIC" field to the corresponding step in the diagram below.

Step 1: Configuring the Service Implementation

The first step in implementing security is when configuring a service implementation. At this step, you specify the identifier of the application on the target card that is to receive the data delivered by the request. For Java cards supporting GSM 03.48, this is a TAR. For ESMS V1 and ESMS V2, it is a folder on the target card.

This information is specified in the service implementation's properties. For example:

Figure 78 - Service Implementation Properties



Or, if batchloading the service implementation:

```
<ServiceImplModule>
    <ServiceImpl
        name = "UpdateUST_Generic_0348_3G"
        serviceName = "Update UST"
        state = "1"
        securityLevel="1"
        applicationIdentifier="B00001"
    ...
    ...
```

When you launch the service request, OTA Manager searches in the card structure of the appropriate card profile for an application with a matching identifier. For example:

```
<CardStructure>
...
<JavaApplet aid="A00000001821001800000000B00001">
    <JavaAppletInstance aid="A00000001821001800000000B00001"
        status="SELECTABLE" estimatedSize="0"
        securityDomain="A000000018434D" tar="B00001"/>
</JavaApplet>
...
</CardStructure>
```

Step 2: Identifying the Target Application

OTA Manager next extracts the AID of the Java applet instance from the card structure area of the card profile and uses it to search in the **EntityProperties** area of the card profile for a matching Java applet instance properties definition. For example:

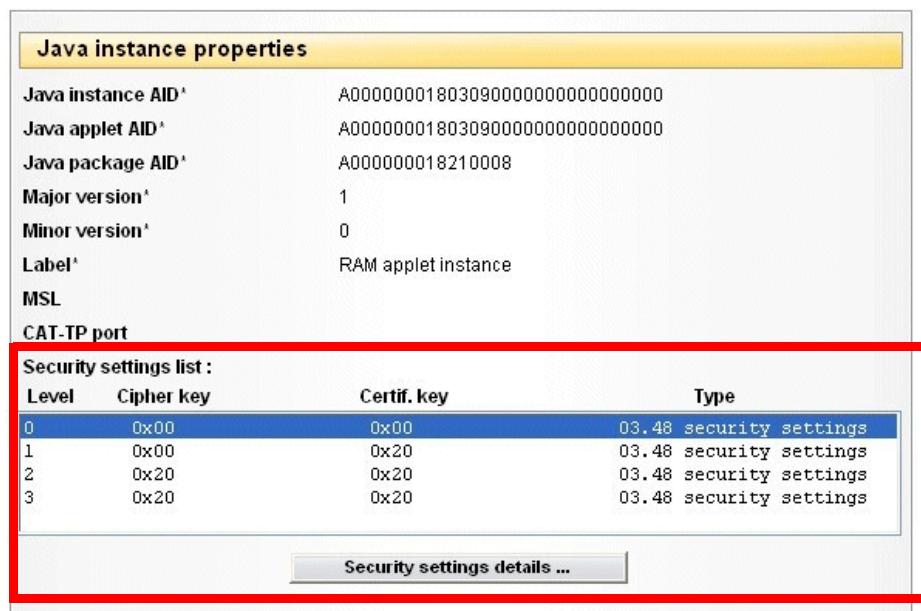
```
<EntityProperties>
...
<JavaAppletInstanceProperties
    instanceAid="A00000001821001800000000B00001"
    appletAid="A00000001821001800000000B00001"
    ...
    </JavaAppletInstanceProperties>
...
</EntityProperties>
...
<CardStructure>
...
<JavaApplet aid="A00000001821001800000000B00001">
    <JavaAppletInstance aid="A00000001821001800000000B00001"
        status="SELECTABLE" estimatedSize="0"
        securityDomain="A000000018434D" tar="B00001"/>
    </JavaApplet>
...
</CardStructure>
```

Step 3: Identifying the Security Mechanisms

OTA Manager next attempts to find a match between the security levels configured in the Java applet instance properties of the card profile and the security level specified in the service implementation properties.

If the security level specified in the service implementation matches one of these security levels, that security level is selected and the security mechanisms specified within the security level are applied:

Figure 79 - Security Levels in the Management Interface



The screenshot shows a management interface window with the following details:

Java instance properties

Java instance AID*	A00000001803090000000000000000000
Java applet AID*	A00000001803090000000000000000000
Java package AID*	A000000018210008
Major version*	1
Minor version*	0
Label*	RAM applet instance
MSL	
CAT-TP port	

Security settings list :

Level	Cipher key	Certif. key	Type
0	0x00	0x00	03.48 security settings
1	0x00	0x20	03.48 security settings
2	0x20	0x20	03.48 security settings
3	0x20	0x20	03.48 security settings

Security settings details ...

Or, when batchloading:

```
<SecurisationSet>
  <Securisation0348 index="4" spi="1E21" kic="10" kid="10" >
  </Securisation0348>
  <Securisation0348 index="3" spi="1621" kic="10" kid="10" >
  </Securisation0348>
  <Securisation0348 index="2" spi="0621" kic="10" kid="10" >
  </Securisation0348>
  <Securisation0348 index="1" spi="0221" kic="00" kid="10" >
  </Securisation0348>
  <Securisation0348 index="0" spi="0000" kic="00" kid="00" >
  </Securisation0348>
</SecurisationSet>
```

Specifying security level 0 in the service implementation would, in this example, select the values “0000” for the two SPI bytes, “00” for the KiC byte, and “00” for the KID byte. See “Security Mechanisms” on page 280 for more details on the security mechanisms themselves and how to specify their use in the security level.

The selected security mechanisms are later implemented by OTA Manager V5.1’s Message Builder module when formatting the SMS messages.

Step 4: Extracting Security Settings

The properties of each applet defined in the structure area of the card profile include a reference to the on-card security domain that handles security settings on behalf of the applet.

In many cases, the security domain is integrated into the interpreter applet, so the reference to the security domain is the AID of the applet instance itself. For example, in the XML used to batchload the card profile, this situation would appear as follows:

```
<JavaAppletInstance aid="A0000000183090000000000B00010"
  status="SELECTABLE" estimatedSize="0"
  securityDomain="A0000000183090000000000B00010"/>
```

In other cases, one or more independent security domains may be present on the card, in which case each has its own unique AID and the system applet includes a reference to the security domain AID instead:

```
<JavaAppletInstance aid="A0000000183090000000000B00010"
  status="SELECTABLE" estimatedSize="0"
  securityDomain="A000000180308"/>
```

When provisioning card instances, you supply values for the security mechanisms implemented by the security domains that are present on the card. In the following example, a security domain with AID A000000180309 manages a single key set:



The actual values of individual keys in each key set, together with the values of synchronization counters associated with the key sets, are provisioned in a key file that is batchloaded directly into the Card Security database.

For example:

```
<SecurityDomain
    securityDomainID="A00000001803090000000000123456"
    defaultSyncID="default"
    implicitRcAlgoNumber="6"
    proprietaryRcAlgoNumber="6" >
    <Sync value="0000000001"/>
    <Keyset versionNumber="1">
        <Sync value="0000000001"/>
        <Kic value="0011223344556677" algoNumber="2"/>
        <Kid value="8877665544332211" algoNumber="2"/>
        <Kik value="1122334455667788" algoNumber="2"/>
    </Keyset>
    <Keyset versionNumber="2">
        <Sync value="0000000001"/>
        <Kic value="00112233445566770011223344556677" algoNumber="3"/>
        <Kid value="88776655443322118877665544332211" algoNumber="3"/>
        <Kik value="11223344556677881122334455667788" algoNumber="3"/>
    </Keyset>
</SecurityDomain>
```

Where:

- **securityDomainID** is the AID of the security domain.
- **versionNumber** is the key set number.
- **value** is the value of a key.

Note: The key values themselves (`value="..."`) may be encrypted using a transport key. Refer to “Transport Keys” on page 77.

-
- **algoNumber** is the identifier of the algorithm to use.

OTA Manager extracts the algorithms and key values from the Card Security database and uses them to format SMS messages.

Security Mechanisms

The various GSM 03.48 security mechanisms are described briefly in the sections that follow.

No Security

Although rarely seen in real-life situations, you are not obliged to implement any security mechanisms in service or campaign requests. In the command SPI byte, you can simply set all bits to zero to indicate that no security is being used. For example, if batchloading security settings:

```
<Securisation0348 index="0" spi="0000" kic="00" kid="00" >
</Securisation0348>
```

When formatted by OTA Manager V5.1, the data in the message remains unencrypted, and the receiving application can retrieve this data without having to invoke the security domain to perform any security checks on the message.

Similarly, you can set all bits of the response SPI byte to zero to indicate that no security mechanisms are to be implemented in the response packet:

```
<Securisation0348 index="0" spi="0000" kic="00" kid="00" >
</Securisation0348>
```

There is obviously an increased threat of malicious attack if no security is implemented.

The Synchronization Counter

A synchronization counter is used to prevent replay attacks and to “re-synchronize” OTA Manager and the SIM card when the OTA message transmission fails for whatever reason.

Synchronization consists in comparing two values; one stored in the CNTR field of the SMS message’s command packet, and a second one stored on the SIM card under the control of a security domain.

Implementing synchronization requires setting the SPI byte value to indicate use of a synchronization counter. The following settings are available:

Table 103 - Specifying User of a Synchronization Counter

Setting	Value to specify in XML
No counter available	spi="0000"
Counter available	spi="0800"
Counter higher	spi="1000"
Counter one higher	spi="1800"

The receiving entity retrieves the value of the synchronization counter from the message and compares it to the value of a synchronization counter on the card. The message is considered to be valid if the counter’s value is greater than the value stored in the SIM card, in which case the local counter is incremented.

The Redundancy Check

A redundancy check is the simplest to implement and least secure of the security mechanisms outlined in the GSM 03.48 specifications. It can only really be used to check that the message was correctly received by the receiving entity.

The implementation of a redundancy check requires several steps:

- 1 Update the SPI value in the security level specified in the service implementation to indicate use of a redundancy check.
- 2 Update the key identifier (KID) byte to indicate the algorithm used to calculate the redundancy checksum.

OTA Manager V5.1 calculates the redundancy checksum and writes it into the RC/CC/DS field of the message during formatting.

For example, if batchloading a card profile:

```
<Securisation0348 index="4" spi="0100" kic="10" kid="01" >
</Securisation0348>
```

In this example, the SPI byte 1 value (“01”, or 00000001 binary) indicates use of a redundancy check, and the KID byte indicates “Algorithm known implicitly by both entities”, meaning that the algorithm set in the field “default implicit algorithm for RC computation” of the corresponding security domain is used.

The Cryptographic Checksum

A cryptographic checksum is a string of bits derived from a secret key, part or all of the message's contents, and possible further information (for example, part of the command header). The secret key must be known to both the sending entity and the receiving entity. The checksum is usually coded on eight bytes (although other values can be used) and is used to check both the integrity and the authenticity of the received message.

The implementation of a cryptographic checksum requires several steps:

- 1 Update the command SPI byte to indicate use of a cryptographic checksum.
- 2 Update the key identifier (KID) byte in the command header to indicate that the algorithm used to calculate the cryptographic checksum is known to both the sending and receiving entity, and set the key set identifier.

OTA Manager V5.1 calculates the cryptographic checksum and writes it into the **RC/CC/DS** field of the command header.

The receiving entity authenticates the message by comparing the content of the CC field extracted from the message with a value computed internally by the SIM card using the same secret key as the sender.

For example:

```
<Securisation0348 index="4" spi="0100" kic="10" kid="10" >
</Securisation0348>
```

In this example, the SPI byte 1 value ("01", or 0000 0001 binary) indicates use of a redundancy check, and the KID byte indicates that key set 1 is to be used and that the "Algorithm known implicitly by both entities", that is, use the security domain's default algorithm.

Ciphering

A cryptographic algorithm, or cipher, is a mathematical function used to encrypt or decrypt the contents of SMS messages. A cryptographic algorithm works in combination with a key—a word, number, or phrase—to encrypt the data contained in the secured header part of the message and parts of the security header. The same key must be known to both the sender and receiver of the message to be decrypted.

Digital signatures are not supported by OTA Manager.

The implementation of a ciphering mechanism requires several steps:

- 1 Update the command SPI byte 1 to indicate the use of ciphering.
- 2 Update the key certificate (KiC) byte to indicate the algorithm and algorithm mode, (if applicable), and identify the key set to use. For example, you could select DES as the algorithm and DES in cipher block chaining (CBC) mode with 2 keys as the ciphering mode, and specify use of key set 2.

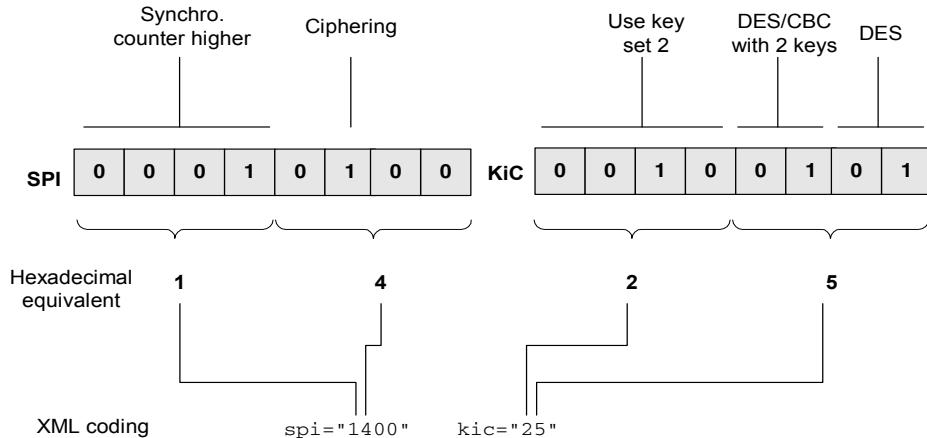
Refer to "Appendix I - GSM 03.48 Message Structure" for implementation details.

The following XML code shows a security level that implements ciphering:

```
<Securisation0348 index="2" spi="1400" kic="25" kid="00">
</Securisation0348>
```

This is illustrated in the following figure:

Figure 80 - Specifying Ciphering in the SPI and KiC Bytes



The receiving card's security domain uses this information to select the algorithm and key set to use to decipher the data.

The Minimum Security Level

On newer Java Cards that support this parameter, you can specify a minimum security level.

The minimum security level (MSL) is an additional security mechanism performed after all other security checks have been successfully performed on the GSM 03.48 header. The targeted application only executes the script contained in the message if the message implements a minimum number of the previously described security mechanisms and, optionally, if the message uses specific key sets for calculating security. If the SMS implements a level of security less from the Minimum Security Level, the command carried within the message is rejected.

The minimum level of security that OTA Manager is to apply to messages sent to an on-card application is specified in the **MSL** parameter of the Java Applet Instantiation File.

Note: Coding of the MSL is described in the 3GPP TS 23.048 (ETSI TS 102 226) specification.

In the following example, the specified MSL is “00”. This means that there is no requirement on the SPI field:

```
<JavaAppletInstanceProperties
    instanceAid="A00000001803090000000000B00010"
    label="RFM applet instance"
    appletAid="A00000001803090000000000B00010"
    packageAid="A0000000180309"
    packageMajorVersion="1"
    packageMinorVersion="0"
    msl="00">
    <SecurisationSet>
        <Securisation0348 index="0" spi="0000" kic="00" kid="00"/>
        <Securisation0348 index="1" spi="0221" kic="00" kid="11"/>
    </SecurisationSet>
</JavaAppletInstanceProperties>
```

All service implementations used to manage the applet must have a security level greater than or equal to the value of the MSL configured for the applet. Check this as follows:

- 1 Check the **Security Level** for the service implementation upon which the service request or campaign is based. This can be viewed on the Service Implementation Information window (**Card Manager > Service**, then select the service implementation and click **View**) as follows:

Figure 81 - Security Level Specified in the Service Implementation

The screenshot shows a software interface titled 'Service implementation information'. It contains several configuration fields:

- Service name: 2G_0348_Update_ADN
- State: Active
- Java implementation class*: rca.serviceImpl.generic0348.Up...
- Script template: rca/serviceImpl/generic0348/Up...
- Security level*: A field containing the value '5'.
- Application target identifier: B00010
- Card content checked: A checked checkbox.
- Card content updated: An unchecked checkbox.
- Transport: A dropdown menu.

At the bottom left, a note says 'All mandatory fields are specified by a *'. At the bottom right are 'Validate' and 'Cancel' buttons.

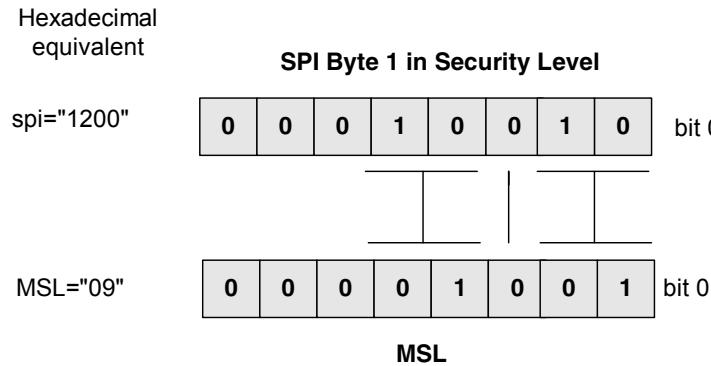
- 2 In the card profile, check the value of the SPI byte 1 of the corresponding security level. In the XML representation of a card profile, for example, this might be:

```
<SecurisationSet>
    <Securisation0348 index="0" spi="1200" kic="00" kid="00" >
    </Securisation0348>
    ...
<SecurisationSet>
```

Where:

- **index** is the security level (0).
- **spi="1200"** means that the SPI byte 1 value is "00010010". This means "use a cryptographic checksum (CC), the synchronization counter value must be higher than the value held on the card, and do not send proof of receipt (PoR)".

- 3 Compare the SPI byte 1 value against the MSL specified in the applet instance properties by performing a bit-wise comparison of the two values. An example, using the SPI byte 1 value of "00010010" and the MSL of "00001001" is shown below:

Figure 82 - Comparing the SPI Byte 1 and the MSL

Specifically, the following comparisons are performed:

- Bits 0 and 1 of the SPI byte 1 are equal to or greater than bits 0 and 1 of the MSL.
- Bit 2 of the SPI byte 1 is equal to or greater than bit 3 of the MSL
- Bits 3 and 4 of the SPI byte 1 are equal to or greater than bits 3 and 4 of the MSL.

In this example, the checks are performed as follows:

- “10” is greater than “01”
- “0” is equal to “0”
- “10” is greater than “01”

The specified security level therefore exceeds the MSL, and the check is successful.

- 4 Check that security data for the applet’s security domain has been provisioned in the card security database, as described in “Chapter 7 - Managing Card Instances”.

Security Domains

A security domain is an on-card application that controls security services, such as key set and synchronization counter handling, encryption, decryption, digital signature generation and verification for applications, on behalf of the card issuer or application provider.

On Java Cards, the Card Manager applet usually incorporates its own security domain. However, more than one security domain can be installed on any card, each managing its own set of security mechanisms.

Security domains are identified by an AID. For example, on GemXplore Xpresso V3 Java cards, the Card Manager applet is the default security domain, and can be addressed either by specifying its TAR (“00 00 00”) or by omitting an AID completely, in which case the Card Manager applet is selected by default.

On the card side, the security domain AID is specified when the applet is provisioned on the card to indicate which security domain is to be used by the applet. This is done by specifying the **SECURITY_DOMAIN_AID** parameter with the **Install (for Install)** command used to instantiate the applet on the card.

Key Sets

Each security domain must contain between one and 15 different key sets. Each key set contains three keys, with the length of the keys determined by the algorithm being used:

Table 104 - Key Sets

Key index	Key label	Cryptographic function
1	KiC	Confidentiality (ciphering)
2	KID	Integrity (cryptographic checksum)
3	KiK	Key Confidentiality

In the GSM 03.48 header:

- Calculation of encryption is performed with KiC key.
- Calculation of a cryptographic checksum (CC) is performed with the KiD key.
- The KiK key is used for decrypting the other keys. It is optional.

By default, all encryption and signature calculation is performed using 3DES.

Handling the Proof of Receipt

A “Proof of Receipt” (PoR) is contained in the response packet sent by the remote SIM card to OTA Manager V5.1. If you select “PoR required” (see “Adding Security Settings” on page 54), the PoR is sent when a message is received successfully or to notify that the delivery failed. If you select "PoR required only on error", the PoR is only sent when the message delivery fails.

A PoR may be requested for any service execution by configuring the security level of the targeted entity to match that of the service implementation. Refer to “Chapter 5 - Managing Services” and “Chapter 6 - Provisioning Card Profiles” for details.

OTA Manager V5.1 cannot correctly manage the PoR when a service invocation generates several command packets that are formatted as sequences of concatenated messages (OTA Manager V5.1 is unable to wait for the PoR of the previous command packet before starting to send the next command packet).

To work around this limitation, if more than one concatenated sequence of messages is required then the PoR should be requested from the card only for the *last* command packet. This is done automatically in all services delivered with OTA Manager V5.1, but must be implemented manually in custom services.

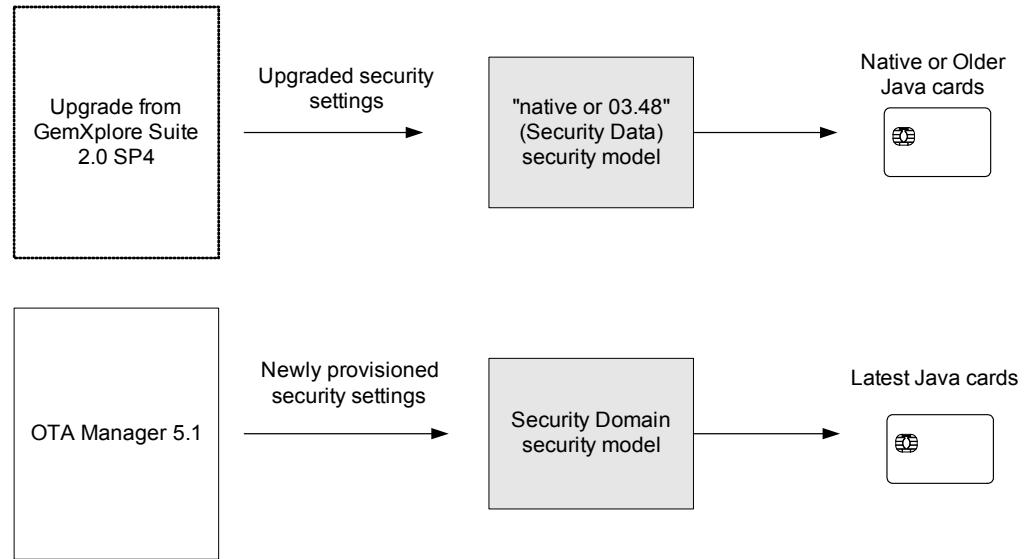
The overall success or failure of the service invocation is therefore determined by examining the PoR of the last command packet.

Backward Compatibility of Security Models

When OTA Manager is installed over an earlier version, all existing security data is marked as belonging to the “native or emulated 03.48” Security Data security model. Backward compatibility is assured for all security settings that have been provisioned on the previously installed platform. Existing formatting capabilities are therefore maintained and continue to function normally for all cards.

The situation is illustrated in “Figure 83”:

Figure 83 - Security Models



GSM 03.48 Message Structure

This appendix describes the contents of the GSM 03.48 secured packet in detail.

GSM 03.48 Secured Packet Structure

Each OTA message is contained within a secured packet. The secured packet can contain either formatted data, that is, data with a predefined structure, or unformatted data, that is, data in a proprietary or non-standard format.

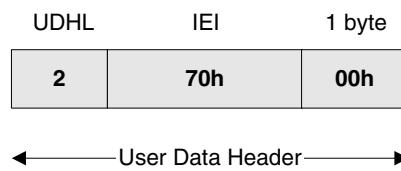
If the TPDU indicates that an unformatted command packet follows, the command packet can contain up to 140 bytes of proprietary data.

Structure of the Command Packet

The command packet is made up of a command header and up to 120 bytes of secured data. To be considered as a GSM 03.48 command packet:

- The UDHI bit (TP-MTI field) in the TPDU must be set.
- The User Data Header (UDH) must be two bytes in length, with an IEI value of “70h” and a data length byte equal to “00h”. If any other value of IEI is found in the UDH, the message is considered as an unformatted SMS message. The structure of the UDH is therefore as shown in “Figure 84”:

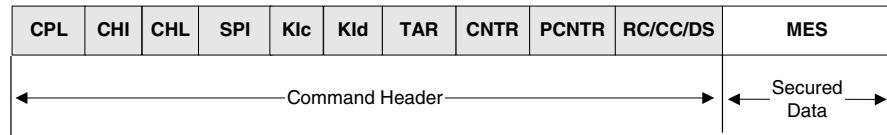
Figure 84 - User Data Header (UDH) Structure



Note: If concatenation is being used, the UDHL field contains “7” and the UDH includes a concatenation header.

The command packet itself is formatted as shown in “Figure 85”:

Figure 85 - Command Packet Structure



Where:

- **CPL.** Command Packet Length, indicates the total number of bytes in the command packet, from the beginning of the Command Header to the end of the Secured Data (including concatenated message data and any padding bytes).
- **CHI.** Command Header Identifier. Identifies the command header. This field is always null (0 bytes) in SMS point-to-point messages.
- **CHL.** Command Header Length, indicates the total number of bytes to the end of the Command Header (either 0Dh, 11h or 15h).
- **SPI.** Secured Packet Information. Indicates, in byte 1 (see “Figure 86”), the security verifications to be made on the command packet, and, in byte 2 (see “Figure 87” on page 291), the settings to be used to build the response packet returned by the responding entity.

Figure 86 - SPI Byte 1 (the Command SPI)

SPI Byte 1

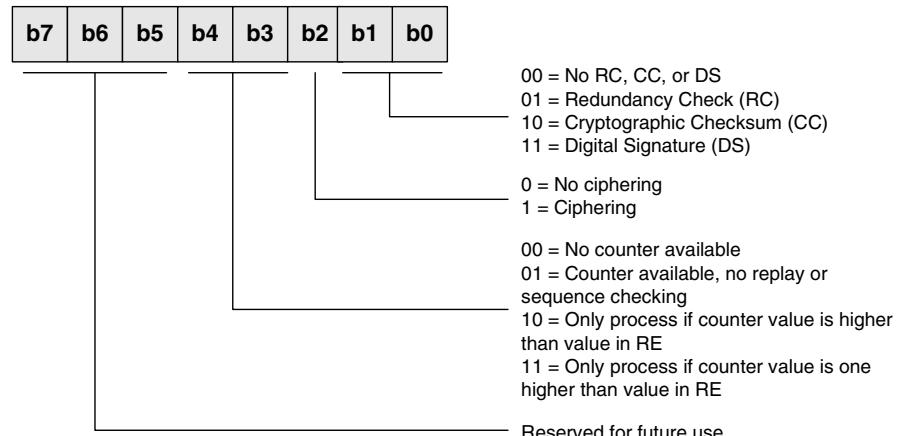
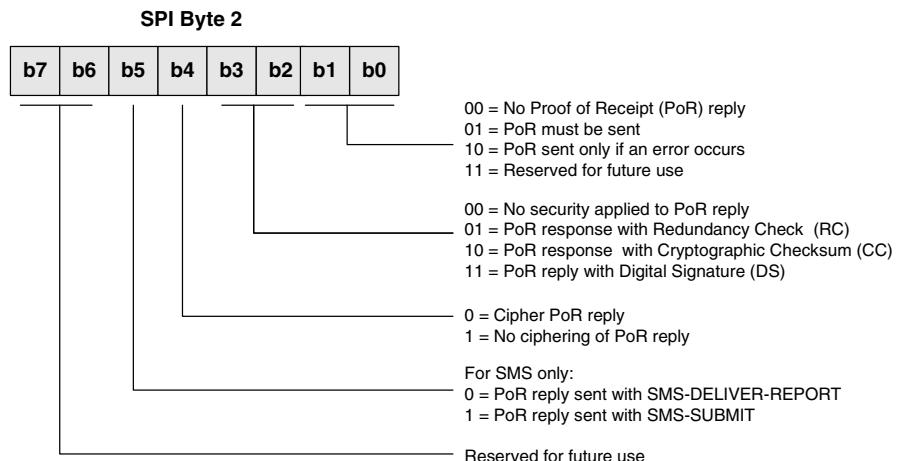
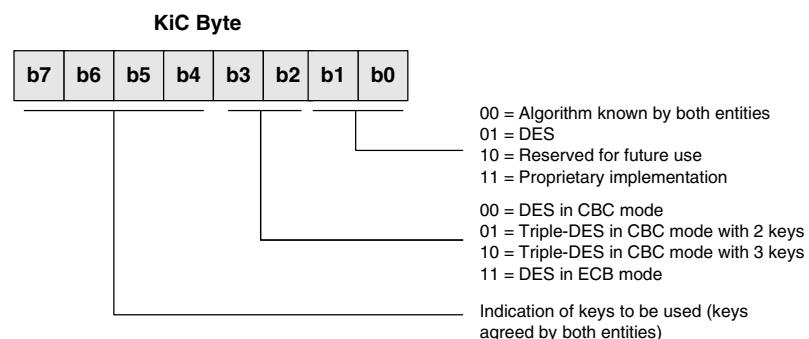


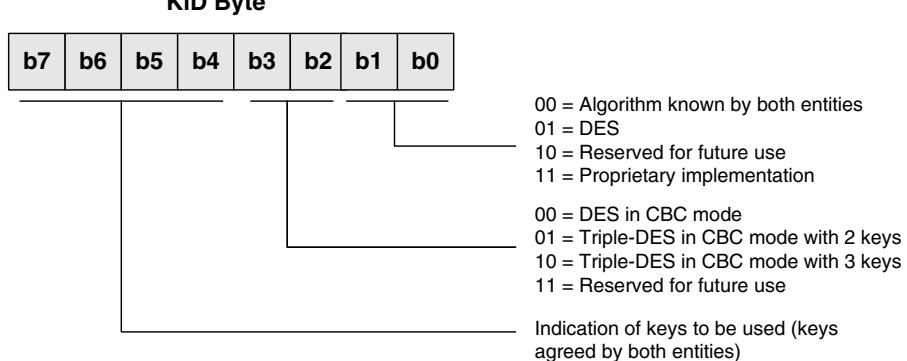
Figure 87 - SPI Byte 2 (the Response SPI)

- **KiC.** Contains information on the ciphering key, algorithm, and key set to be used to decipher the received message and cipher the response packet:

Figure 88 - The KiC Byte

Note: Release 6 of the GSM 03.48 standard specifies that the key set number must be the same in both the KiC and KID for any value other than 0. If the message contains a KiC and KID specifying two different key sets, it will be rejected by the card.

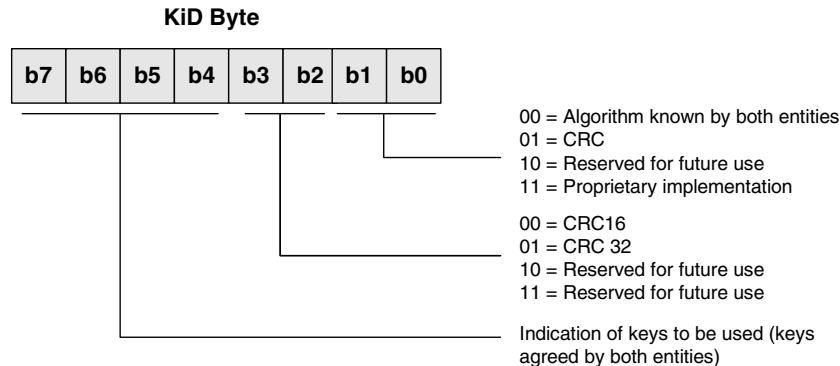
- **KID.** Contains information on the certification algorithm and key set to be used for the cryptographic checksum (CC) or digital signature (DS). The KID byte's format is different for release 5 and 6 of the GSM 03.48 standard:
 - For Release 99 and Release 5:

Figure 89 - The KID Byte (Release 99 and Release 5)

- For release 6, the format of this byte indicates whether the KiD is for a cryptographic checksum or a redundancy checksum.

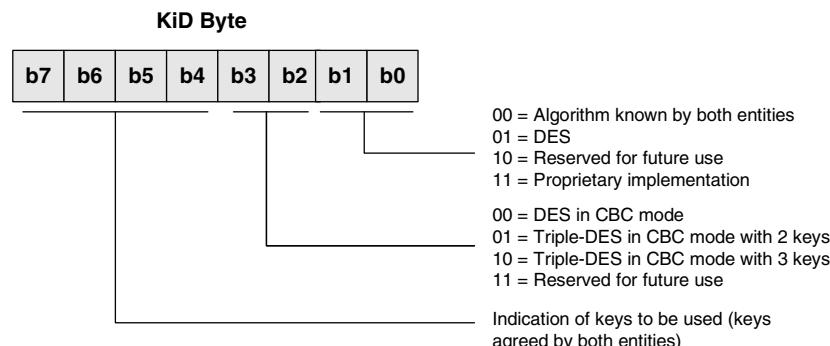
If bits 0 and 1 of the SPI byte 1 (see “Figure 86” on page 290) contain “01” (redundancy checksum), the KID is formatted as follows:

Figure 90 - The KID Byte (Release 6, Redundancy Checksum)



If bits 0 and 1 of the SPI byte 1 (see “Figure 86” on page 290) contain “10” (cryptographic checksum), the KID is formatted as follows:

Figure 91 - The KID Byte (Release 6, Cryptographic Checksum)



Note: Digital certificates are not supported by all SIM card types.

- TAR.** Toolkit Application Reference identifying the applet for which the OTA message is intended.
The TAR identifies the application on the card that is the target for the OTA message. If the TAR is not defined for this applet, the applet cannot be triggered by OTA messages.
- CNTR.** Contains a synchronization counter incremented each time a new OTA application is sent. It re-synchronizes the SIM with the external world when the message transmission fails and is used to prevent replay attacks.
- PCNTR.** Padding counter indicating the number of 00h padding bytes at the end of the secured data.
- RC/CC/DS.** Contains either a redundancy checksum (RC), a cryptographic checksum (CC) or a digital signature (DS).

The RC is simply used to check the integrity of message, and does not involve any secret keys.

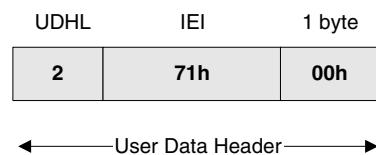
The CC is used to check the authenticity of the OTA message. It is calculated using elements from the secured header and data, and a secret key dedicated to securing remote OTA messages.

Structure of the Response Packet

The response packet is generated by the receiving entity. It could contain information on result execution of the command packet.

The User Data Header (UDH) must be two bytes in length, with an IEI value of “71h” and a data length byte equal to “00h”. If any other value of IEI is found in the UDH, the message is considered as an unformatted SMS message. The structure of the UDH is therefore as shown in “Figure 92”:

Figure 92 - User Data Header (UDH) Structure



Note: If concatenation is being used, the UDHL field contains “7” and the UDH includes a concatenation header.

The response packet itself is formatted as shown in “Figure 93”:

Figure 93 - Response Packet Structure

RPI	RPL	RHI	RHL	TAR	CNTR	PCNTR	Status Code	RC/CC/DS	ADR

Where:

- **RPI.** Response Packet Identifier. Identifies this as a response packet. The RPI field at the GSM 03.48 level is equal to the IEI field in the GSM 03.40 layer.
- **RPL.** Response Packet Length. The number of bytes from the RHI to the end of the ADR (Additional Response Data) field.
- **RHI.** Response Header Identifier. Identifies the response header. For SMS point-to-point messages, this field is always null (0 bytes).
- **RHL.** Response Header Length. Number of octets from the RC/CC/DS field to the end of Response Status Code byte.
- **TAR.** Toolkit Application Reference. This is a copy of the contents of the TAR in the command packet.
- **CNTR.** Synchronization counter for replay detection and sequence integrity counter. This is a copy of the contents of the CNTR found in the command packet.
- **PCNTR.** Padding counter. The number of padding bytes at the end of the ADR (Additional Response Data).

- **Response Status Code.**

Table 105 - Response Status Code Values

Status Code (hexadecimal)	Meaning
00	PoR received successfully.
01	RC/CC/DS failed.
02	CNTR (synchronization counter) less than or equal to the current value (and therefore in error).
03	CNTR higher than the current value (and therefore correct).
04	CNTR blocked (the maximum synchronization counter value was reached).
05	Ciphering error
06	Unidentified security error. This code is for the case where the receiving entity cannot correctly interpret the command header and the response packet is sent unciphered with no RC/CC/DS.
07	Insufficient memory to process incoming message.
08	This status code means "more time". It is used if the receiving entity or application needs more time to process the command packet due to timing constraints. In this case, a further response packet should be returned to the sending entity once processing has been completed.
09	TAR unknown.
0A	Insufficient security level (Release 6 only).
0B - FF	Reserved for future use.

Note: PoRs received with status codes equal to 08 or 0B are ignored by the platform. The request/invocation waits for another PoR until the expiration date is reached.

- **RC/CC/DS.** Redundancy Checksum/Cryptographic Checksum/Digital Signature.
- **ADR.** Additional Response Data. For more details on codes that can be returned see "Additional Data Response Codes" that follows.

Additional Data Response Codes

The Additional Data Response (ADR) provides details about the execution of the command packet on the card.

The codes returned in this field depend both on the commands sent (RAM, RFM, and so on) and on the card vendor version. These codes can be found in ETSI and GlobalPlatform standards (for example, the code 62 F1 is described in the standard ETSI 102.226 *Remote APDU structure for UICC based applications*).

The tables below provide a non-exhaustive list of the codes that occur most frequently.

Table 106 - Application-independent Errors

Additional Data Response (in hexadecimal)	Description
67 00	Incorrect length parameter
6B 00	Wrong P1 or P2 parameter
6D 00	Unknown instruction code
6E 00	Wrong instruction class given in the command
6F 00	Technical problem: ■ Size is lower than authorized ■ File type error ■ Maximum depth reached in the file structure ■ Maximum number of files reached ■ Record length is NULL ■ Creation not allowed ■ Cannot extend EF ■ Rights cannot be established
94 02	Out-of-range record address if an error occurs on P1 or P2 parameter.

Table 107 - Memory Management Errors

Additional Data Response (in hexadecimal)	Description
92 10	Not enough memory
92 20	File identifier already exists
92 40	Memory problem

Table 108 - Referencing Errors

Additional Data Response (in hexadecimal)	Description
94 00	No EF selected
94 02	Possible causes: ■ Out of range ■ File is empty ■ Mode and current pointer not consistent For linear fixed EFs: ■ Mode is PREVIOUS and first record currently selected ■ Mode is NEXT and last record currently selected ■ Mode is CURRENT and no record currently selected ■ Mode is ABSOLUTE and P1 is greater than the number of records in the EF
94 04	Possible causes: ■ File not found ■ Search pattern not found
94 08	File is inconsistent with the command

Table 109 - Message Format Errors

Additional Data Response (in hexadecimal)	Description
6F A2	<p>Possible causes:</p> <ul style="list-style-type: none"> ■ Length of the system field in an executable SMS record is greater than 124 bytes ■ UDL is greater than 140 bytes ■ UDL less than 11 bytes + length of system field

Table 110 - Security Management Errors

Additional Data Response (in hexadecimal)	Description
98 02	<p>Possible causes:</p> <ul style="list-style-type: none"> ■ No CHV code initialized ■ No EF_{KEV} attached to the current file ■ Key with the number specified in the command not found ■ Key not available for internal authentication
98 04	<p>Possible causes:</p> <ul style="list-style-type: none"> ■ Access condition not met ■ CHV verification failed, at least one attempt left ■ Unblock CHV failed, at least one attempt left ■ CHV1 has not been verified successfully beforehand ■ Authentication failed
98 08	<p>Possible causes:</p> <ul style="list-style-type: none"> ■ Not compatible with current CHV status ■ CHV change not allowed. Enable / disable not allowed. ■ CHV is already disabled. CHV is already enabled.
98 10	<p>Possible causes:</p> <ul style="list-style-type: none"> ■ Current file has been invalidated ■ Not compatible with current invalidation status
98 35	Ask Random has not been run beforehand
98 40	<p>Possible causes:</p> <ul style="list-style-type: none"> ■ CHV verification failed, no attempts left ■ UNBLOCK CHV verification failed, no attempts left ■ CHV code blocked. UNBLOCK CHV code blocked.
98 50	Increment cannot be performed. Maximum value is reached.
6F A3	<p>Problem with synchronization counter:</p> <ul style="list-style-type: none"> ■ Message counter less than or equal to the SIM counter ■ The counter difference is greater than the maximum increment value.
6F A4	Problem when checking cryptographic certificate (key not found or cryptography error)
6F A5	Problem with message integrity (incorrect checksum)
6F B0	Problem when updating SIM synchronization counter in EF _{SCRIPTSYSTEM}
6F D3	Incorrect Security Level (OA not checked when SL is LOW)

Table 111 - Kernel and Applicative Command Errors

Additional Data Response (in hexadecimal)	Description
6F B5	Script file not found, invalidated or incorrect file type
6F B7	Execute command forbidden. Script file already in use or access conditions not met.
6F C1	Command execution not authorized
6F 1X	Command decompression error (see Macro errors)
6F BB	Command not supported via the remote channel

Table 112 - Proactive Command Errors

Additional Data Response (in hexadecimal)	Description
6F B5	Script file not found, invalidated or incorrect file type
6F C0	Proactive command request, message execution is aborted
6F C1	Command not allowed (security level not high)
6F C2	Proactive command not supported by the mobile
6F C3	Incorrect length in proactive command
6F C4	Incorrect data in proactive command
6F C5	Permanent problem result from the last proactive command
6F C6	Script aborted by the user
6F C7	Script/mobile capability mismatch

Table 113 - Build Application Menu Command Errors

Additional Data Response (in hexadecimal)	Description
6F D2	Problem when creating the application menu

Table 114 - Card Manager-Specific Errors

Additional Data Response (in hexadecimal)	Description
69 85	Conditions of use not satisfied
69 86	Command not allowed
6A 80	Incorrect parameters in data field (Install command)
6A 84	Not enough memory space
6A 88	Object not found (typically if you try to delete an application that is not on the card)
6F 00	Installation failed (Install command) or link failed (Load command)
6B XX	Incorrect block number in P2 (XX expected)

Table 115 - Remote Management

Additional Data Response (in hexadecimal)	Description
62 F1	The returned MO is more than 140 bytes but concatenation is not activated-supported by the card. In this case, the data are truncated and the rest of the data is lost.

CCI Configuration Properties

This appendix describes the `cci_config.properties` file. This file is generated during the installation procedure using values extracted from the `generic_cci_config.properties` template file and the `FRWKconfig.xml` file. Properties that are configured from `FRWKconfig.xml` are mandatory.

The `cci_config.properties` file is used by the CCI to manage platform functionalities for:

- Connection to the Card Manager (RCA)
- Invocation and campaign parameters
- Reverse proxy
- PoR response option

Property List Description

Card Manager Property

cci.invocationmanager.name

This property specifies the name of the default Card Manager that the CCI connects to.

Format:

```
cci.invocationmanager.name = <frwk_rca_prdname>
```

Where:

`<frwk_rca_prdname>` is the name of the Card Manager defined in `FRWKconfig.xml`.

Request Monitoring Property

cci.monitoring.maxRequests

This property defines the maximum number of requests that can be displayed in request monitoring. It is not configured from `FRWKconfig.xml`.

The default value is 1000.

Format:

```
cci.monitoring.maxRequests = 1000
```

Alphabet Encoding Property

cci.encodingAlphabet.type

This property defines the alphabet encoding used by the platform. Authorized values are ETSI and UCS2. It is not configured from `FRWKconfig.xml`.

The default value is ETSI.

Format:

```
cci.encodingAlphabet.type = ETSI
```

Campaign Scheduling Property

cci.campaign.schedule.destinations.size

This property defines the size (in MB) of the XML file that contains the targets used in campaigns. Possible values are in the range 5 - 70. It is not configured from `FRWKconfig.xml`.

The default value is 5.

Format:

```
cci.campaign.schedule.destinations.size = 5
```

Reverse Proxy Properties

cci.reverseproxy.set

This property defines the reverse proxy activation. Authorized values are 0 (disabled), or 1 (enabled). It is not configured from `FRWKconfig.xml`.

The default value is 0.

Format:

```
cci.reverseproxy.set = 0
```

cci.reverseproxy.name

This property defines the URL of the reverse proxy. It is used when `cci.reverseproxy.set` is set to 1. It is not configured from `FRWKconfig.xml`.

Format:

```
cci.reverseproxy.name = <reverse_proxy_url>.
```

PoR Response Option Property

cci.0348CardSecuritySettings.PoRMgtOption.set

This property defines the PoR management option activation in 0348 Card Security Settings. Authorized values are 0 (disabled), or 1 (enabled). When it is set to 1, the options *PoR response as SMS-SUBMIT* and *PoR response as SMS-DELIVER-REPORT* are visible in 0348 Card Security Settings.

It is not configured from `FRWKconfig.xml`.

The default value is 0.

Format:

```
cci.0348CardSecuritySettings.PoRMgtOption.set = 0
```

Note: This property is only available only if the driver used for invocations or campaigns is the SMPP type and the environment (SMSC, network) allows it.

Campaign Manager Time Zones

The following is a list of time zones that can be specified when starting up the Campaign Manager (see the *OTA Manager V5.1 Installation Guide* for more information).

MIT

Pacific/Apia

Pacific/Niue

Pacific/Pago_Pago

America/Adak

HST

Pacific/Fakaofo

Pacific/Honolulu

Pacific/Rarotonga

Pacific/Tahiti

Pacific/Marquesas

AST

America/Anchorage

Pacific/Gambier

America/Los_Angeles

America/Tijuana

America/Vancouver

PST

Pacific/Pitcairn

America/Dawson_Creek

America/Denver

America/Edmonton

America/Mazatlan

America/Phoenix

MST

PNT
America/Belize
America/Chicago
America/Costa_Rica
America/El_Salvador
America/Guatemala
America/Managua
America/Mexico_City
America/Regina
America/Tegucigalpa
America/Winnipeg
CST
Pacific/Easter
Pacific/Galapagos
America/Bogota
America/Cayman
America/Grand_Turk
America/Guayaquil
America/Havana
America/Indianapolis
America/Jamaica
America/Lima
America/Montreal
America/Nassau
America/New_York
America/Panama
America/Port-au-Prince
America/Porto_Acre
America/Rio_Branco
EST
IET
America/Anguilla
America/Antigua
America/Aruba
America/Asuncion
America/Barbados
America/Caracas
America/Cuiaba
America/Curacao
America/Dominica

America/Grenada
America/Guadeloupe
America/Guyana
America/Halifax
America/La_Paz
America/Manaus
America/Martinique
America/Montserrat
America/Port_of_Spain
America/Puerto_Rico
America/Santiago
America/Santo_Domingo
America/St_Kitts
America/St_Lucia
America/St_Thomas
America/St_Vincent
America/Thule
America/Tortola
Antarctica/Palmer
Atlantic/Bermuda
Atlantic/Stanley
PRT
America/St_Johns
CNT
AGT
America/Buenos_Aires
America/Cayenne
America/Fortaleza
America/Godthab
America/Miquelon
America/Montevideo
America/Paramaribo
America/Sao_Paulo
BET
America/Noronha
Atlantic/South_Georgia
America/Scoresbysund
Atlantic/Azores
Atlantic/Cape_Verde
Atlantic/Jan_Mayen

Africa/Abidjan
Africa/Accra
Africa/Banjul
Africa/Bissau
Africa/Casablanca
Africa/Conakry
Africa/Dakar
Africa/Freetown
Africa/Lome
Africa/Monrovia
Africa/Nouakchott
Africa/Ouagadougou
Africa/Sao_Tome
Africa/Timbuktu
Atlantic/Canary
Atlantic/Faeroe
Atlantic/Reykjavik
Atlantic/St_Helena
Europe/Dublin
Europe/Lisbon
Europe/London
GMT
UTC
WET
Africa/Algiers
Africa/Bangui
Africa/Douala
Africa/Kinshasa
Africa/Lagos
Africa/Libreville
Africa/Luanda
Africa/Malabo
Africa/Ndjamena
Africa/Niamey
Africa/Porto-Novo
Africa/Tunis
Africa/Windhoek
ECT
Europe/Amsterdam
Europe/Andorra

Europe/Belgrade
Europe/Berlin
Europe/Brussels
Europe/Budapest
Europe/Copenhagen
Europe/Gibraltar
Europe/Luxembourg
Europe/Madrid
Europe/Malta
Europe/Monaco
Europe/Oslo
Europe/Paris
Europe/Prague
Europe/Rome
Europe/Stockholm
Europe/Tirane
Europe/Vaduz
Europe/Vienna
Europe/Warsaw
Europe/Zurich
ART
Africa/Blantyre
Africa/Bujumbura
Africa/Cairo
Africa/Gaborone
Africa/Harare
Africa/Johannesburg
Africa/Kigali
Africa/Lubumbashi
Africa/Lusaka
Africa/Maputo
Africa/Maseru
Africa/Mbabane
Africa/Tripoli
Asia/Amman
Asia/Beirut
Asia/Damascus
Asia/Jerusalem
Asia/Nicosia
CAT

EET
Europe/Athens
Europe/Bucharest
Europe/Chisinau
Europe/Helsinki
Europe/Istanbul
Europe/Kaliningrad
Europe/Kiev
Europe/Minsk
Europe/Riga
Europe/Simferopol
Europe/Sofia
Europe/Tallinn
Europe/Vilnius
Africa/Addis_Ababa
Africa/Asmera
Africa/Dar_es_Salaam
Africa/Djibouti
Africa/Kampala
Africa/Khartoum
Africa/Mogadishu
Africa/Nairobi
Asia/Aden
Asia/Baghdad
Asia/Bahrain
Asia/Kuwait
Asia/Qatar
Asia/Riyadh
EAT
Europe/Moscow
Indian/Antananarivo
Indian/Comoro
Indian/Mayotte
Asia/Tehran
MET
Asia/Aqtau
Asia/Baku
Asia/Dubai
Asia/Muscat
Asia/Tbilisi

Asia/Yerevan
Europe/Samara
Indian/Mahe
Indian/Mauritius
Indian/Reunion
NET
Asia/Kabul
Asia/Aqtobe
Asia/Ashgabat
Asia/Ashkhabad
Asia/Bishkek
Asia/Dushanbe
Asia/Karachi
Asia/Tashkent
Asia/Yekaterinburg
Indian/Chagos
Indian/Kerguelen
Indian/Maldives
PLT
Asia/Calcutta
IST
Asia/Katmandu
Antarctica/Mawson
Asia/Almaty
Asia/Colombo
Asia/Dacca
Asia/Dhaka
Asia/Novosibirsk
Asia/Thimbu
Asia/Thimphu
BST
Asia/Rangoon
Indian/Cocos
Asia/Bangkok
Asia/Jakarta
Asia/Krasnoyarsk
Asia/Phnom_Penh
Asia/Saigon
Asia/Vientiane
Indian/Christmas

VST
Antarctica/Casey
Asia/Brunei
Asia/Hong_Kong
Asia/Irkutsk
Asia/Kuala_Lumpur
Asia/Macao
Asia/Manila
Asia/Shanghai
Asia/Singapore
Asia/Taipei
Asia/Ujung_Pandang
Asia/Ulaanbaatar
Asia/Ulan_Bator
Australia/Perth
CTT
Asia/Jayapura
Asia/Pyongyang
Asia/Seoul
Asia/Tokyo
Asia/Yakutsk
JST
Pacific/Palau
ACT
Australia/Adelaide
Australia/Broken_Hill
Australia/Darwin
AET
Antarctica/DumontDUrville
Asia/Vladivostok
Australia/Brisbane
Australia/Hobart
Australia/Sydney
Pacific/Guam
Pacific/Port_Moresby
Pacific/Saipan
Pacific/Truk
Australia/Lord_Howe
Asia/Magadan
Pacific/Efate

Pacific/Guadalcanal
Pacific/Kosrae
Pacific/Noumea
Pacific/Ponape
SST
Pacific/Norfolk
Antarctica/Mcmurdo
Asia/Anadyr
Asia/Kamchatka
NST
Pacific/Auckland
Pacific/Fiji
Pacific/Funafuti
Pacific/Majuro
Pacific/Nauru
Pacific/Tarawa
Pacific/Wake
Pacific/Wallis
Pacific/Chatham
Pacific/Enderbury
Pacific/Tongatapu
Pacific/Kiritimati

References

OTA Transport, Security, and Remote File Management

- 3GPP TS 31.116 (Release 6, V6.4.0): “*Remote APDU Structure for (Universal) Subscriber Identity Module (U)SIM Toolkit applications*”
- ETSI TS 102.226 (Release 6, V6.7.0): “*Remote APDU structure for UICC based applications*”
- 3GPP TS 31.115 (Release 6, V6.3.0): “*Secured packet structure for (U)SIM Toolkit applications*”
- ETSI TS 102.225 (Release 6, V6.4.0): “*Secured packet structure for UICC based applications*”
- 3GPP TS 23.048 (Release 5, V5.8.0): “*Security mechanisms for the (Universal) Subscriber Interface Module (U)SIM Application Toolkit; Stage 2*”.
- 3GPP TS 23.040 (Release 4, V4.4.0) “*Technical realization of the Short Message Service (SMS)*”
- ETSI TS 102.127 (Release 6, V6.0.0) “*Transport Protocol for CAT applications*”.

SIM File System (2G)

- 3GPP TS 51.011 (Release 4, V4.10.0): “*Specification of the SIM-ME interface*”
- GSM 11.11 release 99 v8.2.0: “*Specification of the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface*”

(U)SIM File System (3G)

- 3GPP TS 31.102 (Release 6, V6.6.0): “*Characteristics of the USIM application*”

Others

- ISO/IEC 7816-4: 1995: “*Identification cards—Integrated circuit(s) cards with contacts— Part 4: Interindustry commands for interchange*”
- ISO/IEC 7816-6: 1996 “*Identification cards—Integrated circuit(s) cards with contacts— Part 6: Interindustry data elements*”

Abbreviations

2G	Second Generation
3G	Third Generation
3GPP	Third Generation Partnership Project
ACC	Access Control Class
ADF	Application Dedicated File
ADN	Abbreviated Dialing Number
AID	Application Identifier
API	Application Programming Interface
BCD	Binary Coded Decimal
BDN	Barred Dialing Numbers
BIP	Bearer Independent Protocol
CAP	Card Applet Package
CAT-TP	Card Application Toolkit - Transport Protocol
CBC	Cipher Block Chaining
CC	Cryptographic Checksum
CCA	Customer Care Agent
CCI	Customer Care Interface
CCI	Common Communications Interface
CDR	Call Detail Record
CHL	Command Header Length
CORBA	Common Object Request Broker Architecture
CPL	Command Packet Length
CRC	Cyclical Redundancy Check
CSV	Comma Separated Value
DAP	Data Authentication Pattern
DCS	Data Coding Scheme
DES	Data Encryption Standard
DS	Delivery Status
DS	Digital Signature
DTD	Document Type Definition
EF	Elementary File
ESMS	Enhanced Short Message Service

EST	Enabled Services Table
ETSI	European Telecommunications Standards Institute
FCS	Flow Control Sleep
FPLMN	Forbidden PLMN
GGSN	Gateway GPRS Support Node
GP	GlobalPlatform
GPRS	General Packet Radio Service
GSM	Global System for Mobile communications
GUI	Graphical User Interface
HPLMN	Home PLMN
HPPLMN	Higher Priority PLMN
ICCID	Integrated Circuit Card Identifier
IJC	Interoperable Java Card
IMSI	International Mobile Subscriber Identifier
ISD	Issuer Security Domain
ISO	International Standards Organization
JVM	Java Virtual Machine
m-NFC	mobile-Near Field Communication
MB	Megabyte (1,024 kilobytes or 1,048,576 bytes)
MIB	Management Information Base
MMS	Multimedia Messaging Service
MNO	Mobile Network Operator
MO	Mobile Originated
MRTG	Multi Router Traffic Grapher
MSC	Mobile Switching Center
MSISDN	Mobile Station International Subscriber Directory Number
MSL	Minimum Security Level
MT	Mobile Terminated
NFC	Near Field Communication
NPI	Numbering Plan Identifier
OA	Originating Address
OID	Object Identifier
ONS	Operator Name String
OPLM	Operator-controlled PLMN
ORB	Object Request Broker
OSS	Operational Support System
OTA	Over the Air

PID	Protocol Identifier
PIN	Personal Identification Number
PIX	Proprietary Identifier eXtension
PLMN	Public Land Mobile Network
PMI	Product Management Interface
PoR	Proof of Receipt
POS	Point of Sale
RAM	Random Access Memory
RCA	Remote Card Administrator
RFM	Remote File Management
RHL	Response Header Length
ROM	Read Only Memory
RPI	Response Packet Identifier
RPL	Response Packet Length
SAT	SIM Application Toolkit
SD	Security Domain
SDN	Service Dialing Numbers
SIM	Subscriber Identity Module
SMS	Short Message Service
SMSC	Short Message Service Center
SMSP	SMS Parameters
SNMP	Simple Network Management Protocol
SPI	Secured Packet Information
SPN	Service Provider Name
SQL	Structured Query Language
SS7	Signalling System 7
SST	SIM Service Table
STK	SIM Toolkit
TAR	Toolkit Application Reference
TCP/IP	Transmission Control Protocol/Internet Protocol
TON/NPI	Type of Number/Numbering Plan Identifier
TP	Transport Protocol
TSM	Trusted Service Manager
UDH	User Data Header
UICC	Universal Integrated Circuit Card
UMTS	Universal Mobile Telecommunications Service
URL	Universal Resource Locator

USAT	Universal SIM Application Toolkit
USIM	Universal SIM
UST	USIM Service Table
UTC	Univeral Time Code
UTF	Unicode Transformation Format
WAP	Wireless Application Protocol
WIR	Wired Internet Reader
XML	Extensible Markup Language

Glossary

Applet	A Java application loaded onto a Java card compatible SIM card.
Applet issuer	The person or organization responsible for provisioning and qualifying applets that are to be made available for downloading to SIM cards.
Audit trail	A chronological record of the progress of a transaction that allows an auditor to determine that no errors occurred in the transaction.
Batchloading	The process of provisioning items, such as card definitions or subscriber definitions, by importing data contained in a file directly into the appropriate database.
Bearer	On a GSM network, a communications channel intended for transporting user-related information. With OTA Manager V5.1, communications with remote devices always use the SMS bearer.
Billing ticket	Information generated by a product for use by an external billing application. A typical billing ticket might include the subscriber identifier (MSISDN), the date and time of the transaction, the issuer of the request, the services that were performed, and the number of SMS messages that were sent to the subscriber.
Campaign	The action of downloading, updating, or activating services on a targeted set of cards.
Card contents	See <i>Card instance</i> .
Card definition	A description of the physical characteristics of the card including the chip manufacturer, card operating system and version, and the amount of RAM, ROM and EEPROM memory on the card.
Card instance	An image of the content of a SIM card, stored in the Card Manager instance. OTA Manager V5.1 updates the card instance whenever service requests or campaigns successfully update the corresponding SIM card.
Card issuer	The person or organization responsible for managing subscribers' SIM cards and all operations performed on the cards.
Card profile	A representation of an electrical definition of a SIM card, including the card definition, the file system, pre-installed applets, and related properties.
Card structure	The physical composition of a SIM card in terms of the hierarchical file structure (master files and directory files) and applets.
Card vendor	The manufacturer or reseller of a SIM card.
Channel driver	The component of OTA Manager V5.1 that communicates directly with external systems such as SMSCs or the WIR.
CORBA	A distributed object computing infrastructure that allows applications to work together over networks.

Core API	An application programming interface that allows the development of custom programs and integration with other platforms or third-party systems.
Declaration	See <i>Service declaration</i> .
Delegated management	Operating mode in which the OTA Manager V5.1 platform is run by a Trusted Service Manager (TSM), who securely manages and distributes the service provider's services to the customer base.
Extradition	Prior to GlobalPlatform 2.1, executable load files were associated with a security domain and all applications instantiated from the executable load file were automatically associated with the same security domain. <i>Extradition</i> , introduced with GlobalPlatform 2.1, allows executable load files and applications already associated with one security domain to be associated with a different security domain.
Formatting library	A collection of Java interfaces and classes that generate OTA messages containing secured data.
GlobalPlatform	The GlobalPlatform represents a set of cross-industry technical specifications, which can be used to develop secure and flexible smart card systems. It includes both card and terminal specifications, as well as development tools. Together, these components define an easy-to-use smart card platform upon which standardized applications can be added, such as credit, debit, electronic purse, and loyalty points, as well as applications that support opportunities in a variety of industry segments. The GlobalPlatform works across different cards and operating systems. It enables smart card issuers to choose between operating systems and application developers while providing a core security and card management technology. Originally defined by Visa, the GlobalPlatform has evolved into a cross-industry specification for complete multiple application smart card management. The GlobalPlatform is owned, managed, and developed by GlobalPlatform (www.globalplatform.org).
Group	A collection of related subscribers. A subscriber must be a member of a group and may be a member of more than one. For example, prepaid users or postpaid users.
Implementation	See <i>Service implementation</i> .
Invocation	An instruction corresponding to a single service request targeted at a single target card while it is being processed by the Service Processor module. One invocation may generate multiple SMS messages.
Issuer Security Domain (ISD)	A <i>security domain</i> applet with a wider range of responsibilities than other security domains on the card, for example, loading, installing and deleting applications.

Minimum Security Level	A security mechanism performed after all other security checks have been successfully performed on the GSM 03.48 header. The targeted application only executes scripts contained in messages if the message implements a minimum set of security mechanisms and, optionally, if the message uses specific key sets for calculating security. If the minimum security level is not reached, the command within the message is rejected.
Native	Referring to SIM cards that support SIM Toolkit applications only.
Near Field Communication (NFC)	Short-range, high-frequency, wireless communication technology which enables the exchange of data between devices (for example, a contactless reader and a mobile phone equipped with an NFC card) over short distances (approximately 10 centimeters).
Provisioning	The process of populating OTA Manager V5.1 databases with objects (card profiles, user profiles, group definitions, security settings, and so on).
Remote File Management (RFM)	An OTA protocol enabling you to execute file management APDU commands (for example, SELECT, UPDATE RECORD, DEACTIVATE FILE, or VERIFY PIN) to manage elementary files (EF) on remote SIM cards.
Request	See <i>Service request</i> .
Scenario	A service or set of OTA services to be executed in a specified order as part of a campaign.
Security domain	A special type of applet used to store an application's security data (synchronization counters, cryptographic key sets, and so on). Security domains enable various applications from different providers to share space on a card without compromising the security of any particular provider or application. They also allow the application owner to control its applications without having to share cryptographic keys between entities.
Serial number	A unique identifier of a SIM card, also referred to as the ICCID.
Service	See <i>Service request</i> .
Service declaration	A service's external interface, for example, expected input data, data types, and whether the data is mandatory or optional.
Service implementation	A function (macro language script or Java class) that generates an OTA script containing instructions to be executed by remote SIM cards. For any particular service declaration, several service implementations might be required, each associated with a particular card type.
Service provider	An organization such as a bank, credit card issuer, or transport provider that provides services to mobile subscribers.
Service request	A command to update a SIM card. Each service request is composed of a destination, where you specify which SIM card (mobile phone) or cards to send the request to, the type of service and request parameters, such as whether the request should be sent immediately or not.

Signalling System 7 (SS7)	A global standard that defines the signaling architecture and protocols used by public switched telephone networks.
SIM Toolkit	An ETSI standard for value added services using GSM handsets to perform transactions.
Simple Network Management Protocol (SNMP)	A standard used for gathering statistical data about OTA Manager V5.1 components. SNMP uses management information bases (MIBs), which define what information is available from a particular component.
SMS bearer	The communications channel that carries SMS messages across a network.
SNMP agent	See <i>agent</i> .
Synchronization counter	A security mechanism used to prevent replay attacks. A synchronization counter is incremented each time an OTA message is sent and its value sent with the message. The receiving SIM card increments its own synchronization counter each time a message is received, and compares this to the value in the message.
Timeslot	A configurable period of time during which XCT pre-processing, sending, and post-processing are to be performed.
Toplink	A middleware database mapping tool that caches objects. When using TopLink, a database is only accessed when a Java transaction performs a COMMIT.
Transaction	The action of downloading or updating settings on a single device. Compare with <i>campaign</i> .
Transport key	An optional security mechanism used to protect a batch of cards' security data (such as security keys and synchronization counters) from unauthorized access during transport.
Trap	A message from an SNMP agent indicating a situation that requires immediate attention. You can set a threshold value that determines when the trap is sent.
Trusted Service Manager (TSM)	An organization that provides secure application management capabilities (for example, application downloading, life cycle management) for customers of a mobile network operator on behalf of a service provider.
User	An individual or external system connected and logged on to OTA Manager V5.1. User types are administrator, customer care agent and subscriber.
User account	Based on and linked to a user profile, a user account contains authentication information and specific characteristics of an individual user, such as account status, and expiry date.
User profile	A “template” describing the operations that a particular type of user is authorized to perform. Types of user profiles include administrators, customer care agents and subscribers
Utility	A deck that groups actions or commands shared by several services (for example, Send SMS). This optimizes the space on the SIM by federating commonly-used elements in a single place.

Validity period	A time limit placed on a service or campaign for delivery to the subscriber's SIM card.
Wired Internet Reader (WIR)	Allows a subscriber or customer care agent to access OTA Manager V5.1 services directly over the Internet or Intranet through a card reader connected to a point of sale terminal.

Numerics

- 2G
 - standards **313**
- 3G
 - network **1**
 - standards **313**

A

- account
 - management **8**
 - subscriber, changing password for **8**
 - subscriber, setting up **9**
 - user, setting up **9**
- additional files (SNMP) **208**
- administrator
 - user profile **7**
 - user profile type **7**
- administratorProfile **8**
- agent
 - customer care **8**
 - ORB **219, 223, 224**
 - SNMP **15, 207, 208**
- algo.prop file **85**
- algorithm
 - additional, specifying **85**
 - certification **291**
 - cryptographic **282**
 - decryption **81, 85**
 - default for RC calculation **88**
 - for OTA security **266**
 - in KiC byte **291**
 - in KiD byte **281, 291**
 - key length and choice of **84, 85, 162, 163**
 - known by both entities **83, 85, 281**
 - mode **282**
 - proprietary for RC calculation **88**
 - selecting **283**
 - specifying with **algoNumber** attribute **82, 85, 161, 280**
 - to encrypt security data **77**
- applet instance
 - properties **53**
- assigning an MSISDN **10**
- associating service implementations with profiles **48**

audit trail

- backing up **17**
- configuration **14**
- configuring **14**
- CSV record format **202**
- current status (SNMP) **210**
- DTD **203**
- file format **202**
- in platform monitoring **7**
- parameters **202**
- remote facility **229**
- troubleshooting **228, 269**
- working files, backing up **18**

B

- backing up **17, 19**
- batchloading
 - card definitions **46**
 - card instances **72**
 - card profile **281**
 - card security **81, 85**
 - card security files **86**
 - card structure elements **56**
 - entity definitions **49**
 - group definitions **69**
 - overview of process **128**
 - recommended sequence of **127**
 - security settings **51, 280**
 - service declarations **40**
 - service implementation associations **48**
 - service implementations **40, 277**
 - XML files **127**
- billing
 - activating **32**
 - configuring **15**
 - setting up **203**
 - tickets, generating **32**
- BS_COMPONENT table **113**
- byte
 - application identifier **41**
 - command SPI **280, 282**
 - first SPI **284, 290**
 - key identifier (KID) **281**
 - KiC **52, 279**
 - KID **52, 291**
 - second SPI **291**

C

- C3P0 component **11, 14, 17**
- campaign
 - management **8**
 - managing a **101**
 - monitoring **8**
 - parameters, XCT **107**
 - scheduling XCT **106**
 - security level **284**
 - target files **170**
 - updating content on success **67**
 - XCT, overview **104**
- campaign engine
 - choosing a **101**
 - storage modes **105**
- Campaign Manager
 - how it works **103**
 - services traps **237**
- campaign.dtd **170**
- campaigns
 - monitoring success rate of **123**
- card
 - See *SIM card*
- card content
 - batchloading **71**
 - in Card Manager database **1**
 - overview **68**
 - provisioning **157**
 - searching **72**
 - updating **71**
 - updating (XCT) **105**
 - updating in structure view **88**
 - updating with WIR **93**
 - viewing **73**
- Card database
 - IMSI in **157**
 - managing **2**
 - MSISDN in **157**
 - use by XCT **105**
- card definition
 - batchloading **46**
 - creating **47**
 - DTD **129**
 - files **129**
 - in card profile **43**
 - parameters **130**
 - referencing in card profile **47**
- card instance
 - batchloading **72**
 - managing **70**
 - provisioning a **157**
- Card Manager
 - applet **285**
 - database **41**
 - database, updating card content in (XCT) **105**
 - multiple instances of **28**
- card profile
 - associating service implementations with **42**
 - batchloading **281**
 - card definition in **47**
 - card structure **58, 82, 85**
 - contents of **43**
 - creating **58, 59, 61, 62**
 - DTD **132**
 - file structure **74**
 - GlobalPlatform formatting **62**
 - identifying target applet in **278**
 - link to service implementations **39**
 - linked to card content **72**
 - managing **45**
 - modifying **44**
 - profile independent services **39**
 - provisioning **43**
 - reference to card definition in **47**
 - security level **41**
 - security settings **279**
 - SPI bytes in **284**
 - XML file format **131**
- card record
 - assigning MSISDN **10**
- Card Security database **68, 76**
- card security files
 - batchloading **81, 85, 86**
 - DTD **160**
 - referencing security domain in **57**
- card set **101**
- card structure
 - batchloading **56**
 - deleting elements from **60**
- card vendor
 - batchloading **128**
 - defining **46**
 - definition **43**
 - DTD **128**
- CardStructure** card profile section **58**
- CAT-TP
 - channel driver **33**
 - destination port **174**
 - Java applet properties for **50**
 - messages **1**
 - messaging port **175**
 - port number **51, 54**
 - transport bearer **92**
- CCAPProfile (default CCA profile) **8**
- channel driver
 - configuring for classic campaigns **198**
 - in user profile **34**
 - monitoring **33**
 - status **33**
 - velocity **34**
 - WIR **33**
- channel monitor **33**

cipher block chaining (CBC) **282**
 CMG SMSC
 channel driver **191**
 protocols supported **198**
 CNTR field **281**
 command header
 KID byte **282**
 length (CHL field) **290**
 command packet
 length (CPL field) **290**
 RC/CC calculation for **64**
 security mechanisms in SPI **55**
 structure (03.48) **289**
 synchronization counters **281**
 concatenation **65**
 concrete files **12**
 configuration
 audit trail **14**
 billing **15**
 congestion, monitoring **122**
 Core Framework
 configuring **11**
 memory allocated to **210**
 process file `gemconnect.cfg` **11**
`coreframework.ini` **208, 231**
 creating accounts **7**
 creating subscriber accounts **11**
 cryptographic checksum
 definition of **282**
 in GSM 03.48 standard **291**
 specifying in SPI **55**
 specifying use of in SPI **284**
 CSV (comma separated value) **12, 199**
 custom services **39**
 customer care agent
 default user profile **8**
 user profile **8**

default
 administrator profile **7**
 algorithm for RC **88**
 card profile **45**
 CCA profile **8**
 channel driver **34**
 decryption algorithm **81, 85, 161**
 file contents **57**
 Load command length **63**
 product parameter values **171**
 SMSC **92**
 synchronization counter **88**
 TPUD buffer size **65**
 trace level **30**
 deleting requests **99**
 Delivery Status response message **105, 108, 182**
 destination address, hard coding **102**
 digital signature **282, 291**
 DTD
 audit trail **203**
 campaign.dtd **170**
 card content **72**
 card definition **129**
 card profile **132**
 card security **160**
 card vendor **128**
 cardframework.dtd **129**
 definition **128**
 for service declarations **40**
 group definition **167**
 log file **200**
 messageBuilder.dtd **79, 85**
 MIB definitions **208**
 service declaration **154**
 service implementation **154**
 volume.dtd **167**

D

database
 card instances **82, 85**
 Card Manager **41**
 Card Security **76**
 contents, protecting **68**
 files, backing up **18**
 files, purging **19**
 Framework **14**
 storage mode **18**
 XCT **105**

E

emulated 03.48 security data **81, 85**
 enciphering **282**
 Error facility, configuring **17**
 ESMSV1 **52**
 ESMSV2 **52**

F

FLOW_CONTROL product parameter **107, 109, 110**
 formatting libraries **35**
 formatting properties, Global Platform **62**
 Framework database **14**

G

GemXplore 3G **45, 285**

Global Platform
formatting properties **62**

GlobalPlatform
formatting properties **62**
GOP interpreter applet **49**
grace period (XCT) **108**
group definition

batchloading **69**
DTD **167**

groups
in billing tickets **203**
managing **68**

GROUPS_NBR installation parameter **69**
GSM 03.48

message structure **289–294**
secured packets **289**
security mechanisms **281**

GSM file system **281**

GSM interpreter applet **49**

gxs.additional.file.location property **208**
gxs.xml.file.location property **208**

H

heap size **116**

I

ICCID **10**

IMSI
associated with card instance **67**
in card content **157**
in group definitions **70**

initial card content **44**

InitString **187**

INQUIRY_ACTIVATED product parameter **110**

installation parameters

 GROUPS_NBR **69**

interpreter applets **41, 49**

Invocation Registry **11**

Invocation registry **98, 218**

INVOCATION_CACHE_SIZE product parameter
112

invocations **19, 105, 177**

invocations (XCT) **105**

J

Java applet
example **159**
example XML definition **49**
installation parameters **285**
RAM properties **65**
security properties **49**
security settings for **34**

Java Card

Card Manager applet **285**
entity definitions **49**
properties, applet instance **49**
security model for **81, 85**
security settings for **83, 84, 85**
support for MSL parameter **283**
See also *GemXplore 3G*

K

KiC byte

algorithm in **282**
security level in **279**
specifying key set in **55**
specifying security in **52**
translating to hexadecimal **52**

KiC/KID settings **55**

KID byte

algorithm for RC in **281**
encoding in card profile **52**
format **291**
format (GSM 03.48) **292**
in security data **83, 85**
in security settings **52**
key set identifier in **282, 291**
key set index **286**
redundancy check specified in **281**
specifying security level in **279**

killing, a product **28, 211**

L

Library V3 **35, 82, 85**

Library V4 **35**

life cycle

 product **25**
 request **94**
 requests **94**

log file

 configuration **12**
 DTD **200**

 naming convention **13**

logging facility

 configuring **12**
 log file format **199**
 OID value **207**
 SNMP traps **226**

logging facility, remote

 SNMP traps **227**

M

Management Information Base (MIB)

See *MIB*

- managing
 campaigns **101**
 card instances **70**
 card services **39**
 cards **2**
 groups **68**
 master file **208**
 memory footprint, XCT **112**
 Message Builder **1**, **35**
 messageBuilder.dtd **79**, **85**
 messages, point-to-point **290**, **293**
 MIB
 additional files **16**, **208**
 DTD **208**
 location of **16**
 master file **208**
 monitoring platform with **16**
 structure **207**
 MIN_IN_MEMORY_TIME product parameter **118**
 minimum security level
 See *MSL*
 Mobile Challenge feature **11**
 model, security **275**
 monitor, channel **33**
 monitoring
 campaign counters **216**
 campaigns **8**
 channel drivers **33**
 problems, fixing **269**
 remote, using SNMP **15**
 request, by CCA/subscriber **94**
 requests **94**, **262**
 SNMP traps and counters **207**
 target counters **217**
 throughput and congestion **122**
 monitoring requests **263**
 MSISDN
 group definition with **70**
 in billing ticket **203**
 in card content **157**
 in Card database **157**
 retrieving from Card Manager database **1**
 SIM card link **10**
 MSL
 comparing with SPI byte **284**
 Java Card support for **283**
 security check **283**
 Multi Router Traffic Grapher (MRTG) **113**
 multiple instances, of Card Manager **28**
- N**
- naming convention, log file **13**
 native security settings **52**
 Nokia SMSC
 channel driver **191**
- O**
- oggining **209**
 ORB agent
 port number **219**
 ORB agent port **219**, **223**, **224**
 ORIGINATING_ADDRESS product parameter
111
 OTA system file **281**
- P**
- password, length of mobile challenge **11**
 planning templates **106**
 platform
 monitoring **7**
 point of sale (POS) **33**
 point-to-point messages **290**, **293**
 port
 CAT-TP **51**, **54**
 CAT-TP destination **174**
 CAT-TP messaging **175**
 of host machine for logging **14**
 ORB agent **219**
 SMSC IP **188**
 SNMP Manager **16**
 private key **282**
 product
 life cycle **25**, **211**
 product parameters
 Card Manager (RCA) **173**–**184**
 changing **171**
 exporting and importing **172**
 FLOW_CONTROL **107**, **109**, **110**
 INQUIRY_ACTIVATED **110**
 INVOCATION_CACHE_SIZE **112**
 MIN_IN_MEMORY_TIME **118**
 modifying **31**
 ORIGINATING_ADDRESS **111**
 SENDING_PAUSE_MODE_FINISH_INVOCAT
 IONS **107**
 SMS_CACHE_SIZE **112**
 SMS_PUSH_ALPHA_ID **174**, **252**
 STACK_SIZE **109**
 TP_DA_TON_NPI (XCT) **111**
 TP_OA (XCT) **111**
 XCT **185**–**186**
 production space **12**, **13**
 profile independent services **39**
 provisioning
 card content **157**
 card profiles **43**
 card structure **44**
 files, backing up **18**
 files, purging **19**
 problems, solving **268**
 public key **282**

purging
 database files 19
 invocations 19
 provisioning files 19
 working files 19

Q

quality of service 34
 query, defining a 96

R

RAM (memory) 43
 RAM (remote applet management)
 applets, concatenation option 65
 interpreter applets 49
 RC
 configuring fields to use in calculation 64
 default algorithm 88
 security mechanism 281
 specifying in command packet 55
 specifying in response packet 56
 RC/CC/DS field 282
 redundancy check
 See *RC*
 remote file management
 See *RFM*
 report file
 batchloading 128
 for group creation 69
 location of 17, 177
 request
 life cycle of 94
 monitoring 94
 See also *service requests*
 submitting a 91
 Request acknowledgement 96, 263
 request management 98
 requests
 defining query 96
 deleting 99
 life cycle 94
 searching for 95
 response packet
 identifier (RPI) 293
 length (RPL) 293
 structure (GSM 03.48) 293
 retries
 maximum number of (XCT) 108
 setting number of 176
 retry delay 121
 RFM
 applets 49

root domain, of trace 30
 router, SS7 108

S

search
 requests 95
 secured packet information (SPI) 290
 secured packets 289
SecurisationSet element 50
 security
 card, managing 76
 check, using MSL 283
 database 76
 level 41
 model 275
 properties, of on-card entities 49
 specifying in service implementation 284
 Security Data security model 81, 85
 security domain 44, 57
 Security Domain security model 81, 85
 security level 284
 security mechanisms
 configuring additional 85
 description of 280–283
 in card profile 43
SecurisationSet element 50
 selecting 55
 security model
 backward compatibility of 287
 in Card Security files 160
 Security Domain, implementing 275–286
SecurityData element 160
SecurityDomain element 160
 when batchloading card security 81, 85
 when viewing card security 87
 security settings
 adding 54
 and formatting libraries 34
 batchloading 49, 51, 280
 extracting from card profile 279
 GSM 03.48 51
 native, ESMS 52
 of card instances 67
 sensitive nature of 68
 supported by Library V4 35
 updating using CCI 53
 viewing 87
SECURITY_DOMAIN_AID parameter 285
SecurityData element 160
SecurityDomain element 160
SENDING_PAUSE_MODE_FINISH_INVOCATION product parameter 107

service
 custom **39**
 information **1**
 management **39**
 name, in XML **40**
 profile independent **39**
 quality of **34**
 repository **39**
Update ADN 41
Update MSISDN 10
 value-added **39**
 service declaration
 batchloading **40**
 description of **39**
DTD 154
 service implementation
 batchloading **40, 277**
 description of **39**
DTD 154
 parameters **155**
 service requests
 authorized in user profile **94**
 submitting **91**
 using the WIR **91**
 services traps
 campaign manager module **237**
 monitoring **237**
SIM Ack message 121
SIM card
 concatenation support **64**
 destination address in **102**
 electrical profile **43, 67**
 issued by customer care agent **10**
 link to MSISDN **10**
 list **73**
 manufacturer **43**
PoR 108, 286
 updating with WIR **33**
 updating, typical use case **1**
 vendor **43**
 See also *GemXplore 3G*
SMPP SMSC
 protocol supported **198**
SMS_CACHE_SIZE product parameter 112
SMS_PUSH_ALPHA_ID product parameter 174, 252
SMSC
 and channel drivers **33**
CMG 191, 198
 default **92**
 expiration time **107**
 initialization string (InitString) **187**
 IP port number **188**
 mode, validity period (XCT) **107**
Nokia 191
 properties in requests **92**
SMPP 198
 transmission mode **102, 182**
 types, supported **187**
 validity period (XCT) **107**
SNMP
 agent **15, 207, 208**
 configuration **15**
 remote monitoring **15**
SNMP counters
 target counters **217**
 traffic flow counters **217**
SNMP Manager
 port **16**
SPI
 first byte, format of **290**
 second byte **291**
SPI command packet 55
SPI response packet 56
SS7 105
SS7 mode 108, 110
STACK_SIZE product parameter 109
 starting products **28**
 stopping products **28**
 storage mode
 database **18**
 of campaign engine **105**
 storage space **12**
 structure view
 updating card content in **88**
 submitting service requests **91**
 subscriber
 user profile **8**
 subscriber list
 example **168**
 synchronization counter
 default **88**
 in KID byte **292**
 purpose of **281**

T**TAR**

- in GSM 03.48 standard **292, 293**
- in security model **277**
- in service implementation parameters **155**
- including in RC/CC calculation **64**
- of Card Manager applet **285**
- of destination application **292**
- specifying in card profile **57**
- specifying in service implementations **41**

throughput, monitoring **122****timeslots (XCT)** **106****TP_DA_TON_NPI product parameter (XCT)** **111****TP_OA product parameter (XCT)** **111****TP-DA** **111****trace level**

- default **30**
- editing **30**
- editing for Message Builder **234**

in log file **199**root domain **30****traffic flow counters**monitoring **217****TRANS transmission mode** **102****transmission mode**

- of SMSC **102, 182**
- setting **182**

transport keys **78, 269, 280**creating **78****traps**See *SNMP***troubleshooting****audit trail** **228, 269****executing services** **266****monitoring** **269****sending SMS messages** **262****startup problems** **261****tuning****XCT campaigns** **104****U****update**card content **71****Update ADN service** **2, 39****Update FDN service** **2****user account**managing **7**relation to user profile **7****user data header (UDH)**in GSM 03.48 **289, 293****user profile**

- channel driver management, authorizing **187**
- default installed **8**
- managing **7**
- relation to user account **7**
- requests authorized in **94**
- setting up **8**
- specifying channel driver in **34**
- types **7**

V**validity period**setting (XCT) **107****velocity, of channel driver** **34****viewing card content** **73****volume.dtd** **167****W****WIR****bearer type** **205****channel** **33****channel monitor, activating** **176****submitting requests with** **93****updating card content** **93****using the** **93****using with service requests** **91****working files, purging** **19****X****XCT****campaign scheduling** **106****campaign tuning** **104****Card database use** **105****card sets** **101****database** **105****grace period** **108****memory footprint, calculating** **112****parameters, setting** **107****planning templates** **106****target file** **170****timeslots** **106****validity period, for SMSC** **107****XML files****batchloading** **127**

