

# BONDI

## BONDI – ARCHITECTURE & SECURITY APPLICATION LIFECYCLE EVENTS USE CASES

**VERSION:** Version 1.0  
**STATUS:** Approved Release  
**DATE OF LAST EDIT:** 26<sup>th</sup> May 2009  
**OWNER:** OMTP Limited



# CONTENTS

<b>1</b>	<b>INTRODUCTION .....</b>	<b>5</b>
1.1	DOCUMENT PURPOSE .....	5
1.2	INTENDED AUDIENCE .....	5
<b>2</b>	<b>USE CASES .....</b>	<b>6</b>
2.1	WIDGET RESOURCE INSTALLATION .....	6
2.1.1	<i>Manual Installation from a web server.....</i>	<i>6</i>
2.1.2	<i>Remotely initiated Installation from a web server.....</i>	<i>10</i>
2.1.3	<i>Manual Installation from local storage.....</i>	<i>13</i>
2.2	WIDGET RESOURCE SETTINGS MANAGEMENT .....	16
2.2.1	<i>Auto-start at Terminal boot-up .....</i>	<i>16</i>
2.3	WIDGET RESOURCE DE-INSTALLATION .....	19
2.3.1	<i>Remotely initiated Widget Resource de-installation .....</i>	<i>19</i>
2.3.2	<i>Manual Widget Resource de-installation.....</i>	<i>22</i>
2.4	WIDGET PROCESSING .....	24
2.4.1	<i>Basic Widget Processing .....</i>	<i>24</i>
2.4.2	<i>Widget Processing with network coverage.....</i>	<i>27</i>
2.4.3	<i>Widget Processing without network coverage.....</i>	<i>29</i>
2.4.4	<i>Processing of multiple Widgets.....</i>	<i>30</i>
2.4.5	<i>Widget Engine Execution .....</i>	<i>32</i>
2.4.6	<i>Installed Widget Processing.....</i>	<i>35</i>
2.4.6.1	<i>Start-up of installed Widget Resources during Terminal boot-up.....</i>	<i>35</i>
2.4.6.2	<i>Start-up of an installed Widget Resource by an application.....</i>	<i>37</i>
2.4.6.3	<i>Manual start-up of an installed Widget.....</i>	<i>39</i>
2.4.7	<i>Non-installed Widget Processing .....</i>	<i>41</i>
2.4.7.1	<i>Non-installed Widget Processing from a web server.....</i>	<i>41</i>
2.4.7.2	<i>Non-installed Widget Processing from a local storage.....</i>	<i>44</i>
2.5	WIDGET PROCESSING TERMINATION .....	46
2.6	WIDGET RESOURCE VERSION UPDATE .....	47
2.6.1	<i>Installed Widget Resource version update.....</i>	<i>48</i>
2.6.1.1	<i>Automatic version update .....</i>	<i>48</i>
2.6.1.2	<i>Remotely initiated version update .....</i>	<i>53</i>



2.6.1.3	Manual version update .....	57
2.6.2	<i>Non-installed Widget Resource version update</i> .....	61
2.6.2.1	Automatic version update .....	61
2.6.2.2	Manual version update .....	65
2.6.3	<i>Automatic version update in Roaming Mode</i> .....	69
2.7	WEB PAGE PROCESSING.....	71
<b>3</b>	<b>DEFINITION OF TERMS</b> .....	<b>74</b>
3.1	MAPPING TO OTHER SDOS' TERMS .....	79
<b>4</b>	<b>ABBREVIATIONS</b> .....	<b>79</b>
<b>5</b>	<b>REFERENCED DOCUMENTS</b> .....	<b>81</b>



The information contained in this document represents the current view held by OMTP Ltd. on the issues discussed as of the date of publication.

This document is provided “as is” with no warranties whatsoever including any warranty of merchantability, non-infringement, or fitness for any particular purpose. All liability (including liability for infringement of any property rights) relating to the use of information in this document is disclaimed. No license, express or implied, to any intellectual property rights are granted herein.

This document is distributed for informational purposes only and is subject to change without notice. Readers should not design products based solely on this document.

© 2009 OMTP Ltd. All rights reserved. OMTP and OMTP BONDI are registered trademarks of OMTP Ltd.

# 1 INTRODUCTION

## 1.1 *DOCUMENT PURPOSE*

This document describes the use cases for the life cycle management of Widget Resources. This includes not only the Installation, De-installation and version update, but also the processing of installed and non-installed Widget Resources as well as the rendering of Web Pages. The document complements the set of use cases for the use of mobile Terminal specific functions by Widgets and Web Pages that are defined by the “BONDI Interfaces” task.

The use cases in this document are used in the “BONDI Architecture & Security” task to define functional requirements. They also describe the architectural framework for the security concept which needs to fit into this framework.

## 1.2 *INTENDED AUDIENCE*

This document is targeted at operators, manufacturers and to application developers.

## 2 USE CASES

### 2.1 WIDGET RESOURCE INSTALLATION

#### 2.1.1 MANUAL INSTALLATION FROM A WEB SERVER

<b>USE CASE NAME</b>	Widget Resource Installation from a web server	
<b>SHORT DESCRIPTION:</b>	This use case describes how a user transfers a Widget Resource from a web server to his/her Terminal's local storage and persistently installs it in the Terminal.	
<b>LIFE CYCLE EVENT DESCRIPTION</b>		
<b>CUSTOMER BENEFIT:</b>	The Installation of a Widget Resource allows the use of a Widget after each boot-up of the Terminal without the need to re-download the Widget Resource from a web server again.	
<b>ACTOR(S):</b>	<ul style="list-style-type: none"> <li>• User</li> <li>• Widget Download Application</li> <li>• Widget Engine</li> <li>• Widget Manager</li> <li>• Widget Resource Provisioning Server</li> </ul>	
<b>PRECONDITIONS:</b>	<ul style="list-style-type: none"> <li>• The Widget Resource is provided for downloading on a Widget Resource Provisioning Server</li> <li>• The Terminal is able to communicate via TCP/IP with the Widget Resource Provisioning Server</li> <li>• The Widget Download Application is installed on the Terminal</li> <li>• The Widget Engine is installed on the Terminal</li> <li>• The Widget Manager is installed on the Terminal</li> </ul>	
<b>WORKFLOW (ACTIVITY SEQUENCE):</b>		
	<b>ACTOR INTENTION</b>	<b>SYSTEM RESPONSIBILITY</b>

	<p>1. The user initiates the launch of the Widget Download Application on the Terminal</p>	<p>2. The Widget Download Application is loaded and executed on the Terminal</p>
	<p>3. The user enters the URL of the Widget Resource on the Widget Resource Provisioning Server into the Widget Download Application and requests retrieval of this Widget Resource</p>	<p>4. The Widget Download Application retrieves the Widget Resource from the Widget Resource Provisioning Server</p>
		<p>5. The Widget Download Application discovers that the downloaded resource is a Widget Resource and asks the Terminal to handle the Widget Resource by the default application that is registered for handling Widget Resources</p>
		<p>6. The Widget Manager takes over the Widget Resource and starts the Installation procedure</p>
		<p>7. The Widget Manager generates a prompt on the Terminal UI asking the user to proceed with the Widget Resource Installation, to process the Widget without Installation<sup>1</sup> or to cancel the Installation procedure</p>
	<p>8. The user confirms to proceed with the Widget Resource Installation</p>	<p>9. The Widget Manager proves that the Widget Resource's zip archive and its file entries conform to the W3C Widget specification [2]</p>

<sup>1</sup> The Widget Processing without prior Installation is described in section 2.4.7

		10. The Widget Manager stores the Widget Resource in the Terminal's default Widget Resource repository
		11. The Widget Manager adds the Widget Resource to the Widget Resource List
		12. The Widget Manager stores the Widget Resource's Widget Resource Time Stamp as provided by the Widget Resource Provisioning Server
		13. The Widget Manager generates a prompt on the Terminal UI informing the user about the successful Installation and asking to proceed with the Widget Processing
Variant A		
	14. The user confirms to proceed with the Widget Resource processing	15. The Widget Manager asks the Widget Engine to process the Widget Resource
		16. The Widget Engine processes the Widget Resource as described in the use case "Basic Widget Processing" in section 2.4.1
		17. The Widget Manager completes the Installation procedure
Variant B		
	14. The user closes the prompt	15. The Widget Manager completes the Installation procedure

<b>POST CONDITIONS:</b>	<ul style="list-style-type: none"><li>• The installed Widget Resource is persistently stored in the Terminal</li><li>• The updated Widget List is persistently stored in the Terminal</li><li>• The Widget Resource Time Stamp is persistently stored in the Terminal</li></ul>
<b>APPLICABILITY:</b>	
<b>EXCEPTIONS:</b>	None
<b>REFERENCES:</b>	

**2.1.2 REMOTELY INITIATED INSTALLATION FROM A WEB SERVER**

<b>USE CASE NAME</b>	Widget Resource Installation from a web server initiated by a server-push
<b>SHORT DESCRIPTION:</b>	This use case describes how a remote service initiates the transfer of a Widget Resource from a web server to a Terminal and persistent Installation in the Terminal.
<b>LIFE CYCLE EVENT DESCRIPTION</b>	
<b>CUSTOMER BENEFIT:</b>	<ul style="list-style-type: none"> <li>• The Installation of a Widget Resource allows the use of a Widget after each boot-up of the Terminal without the need to re-download the Widget Resource from a web server again.</li> <li>• The remotely initiated Widget Resource Installation enables a user to use more convenient means than the Terminal to choose a Widget Resource on the Widget Resource Provisioning Server and to ask for the Installation on his/her Terminal.</li> </ul>
<b>ACTOR(S):</b>	<ul style="list-style-type: none"> <li>• Remote Installation Service</li> <li>• User</li> <li>• Widget Engine</li> <li>• Widget Manager</li> <li>• Widget Resource Provisioning Server</li> </ul>
<b>PRECONDITIONS:</b>	<ul style="list-style-type: none"> <li>• The Widget Resource is provided for downloading on a Widget Resource Provisioning Server</li> <li>• The Terminal is able to communicate via TCP/IP with the Widget Resource Provisioning Server</li> <li>• The Remote Installation Service is able to communicate with the Terminal</li> <li>• The Widget Engine is installed on the Terminal</li> <li>• The Widget Manager is installed on the Terminal</li> <li>• The Widget Manager is registered in the Terminal to handle push messages for Widget Resource Installations</li> </ul>
<b>WORKFLOW (ACTIVITY SEQUENCE):</b>	

ACTOR INTENTION	SYSTEM RESPONSIBILITY
1. The Remote Installation Service sends an Installation request to the Terminal to initiate the Installation of a specific Widget Resource that is provided on the Widget Resource Provisioning Server	2. The Terminal discovers that the received request is to be handled by the Widget Manager and asks the Widget Manager to handle the request
	3. The Widget Manager takes over the Installation request and starts the server-push Installation procedure
5. The user confirms the server-push Installation	4. The Widget Manager generates a prompt on the Terminal UI asking the user whether he/she wants to proceed with the server-push Installation of the Widget Resource or to cancel the server-push Installation
	6. The Widget Manager obtains the Widget Resource URL from the Remote Installation Service's Installation request
	7. The Widget Manager sends a request to the Widget Resource Provisioning Server to retrieve the Widget Resource that is referenced by the Widget Resource URL
	8. The Widget Resource Provisioning Server responds with the requested Widget Resource

		9. The Widget Manager proves that the Widget Resource's zip archive and its file entries conform to the W3C Widget specification [2]
		10. The Widget Manager stores the Widget Resource in the Terminal's default Widget Resource repository
		11. The Widget Manager adds the Widget Resource to the Widget Resource List
		12. The Widget Manager stores the Widget Resource's Widget Resource Time Stamp as provided by the Widget Resource Provisioning Server
		13. The Widget Manager generates a prompt on the Terminal UI informing the user about the successful Installation and asking to proceed with the Widget Processing
	Variant A	
	14. The user confirms to proceed with the Widget Resource processing	15. The Widget Manager asks the Widget Engine to process the Widget Resource
		16. The Widget Engine processes the Widget Resource as described in the use case "Basic Widget Processing" in section 2.4.1
		17. The Widget Manager completes the Installation procedure
	Variant B	

	14. The user closes the prompt	15. The Widget Manager completes the Installation procedure
<b>POST CONDITIONS:</b>	<ul style="list-style-type: none"> <li>• The installed Widget Resource is persistently stored in the Terminal</li> <li>• The updated Widget List is persistently stored in the Terminal</li> <li>• The Widget Resource Time Stamp is persistently stored in the Terminal</li> </ul>	
<b>APPLICABILITY:</b>		
<b>EXCEPTIONS:</b>	None	
<b>REFERENCES:</b>		

**2.1.3 MANUAL INSTALLATION FROM LOCAL STORAGE**

<b>USE CASE NAME</b>	Widget Resource Installation from local storage
<b>SHORT DESCRIPTION:</b>	This use case describes how a user installs a Widget Resource from a Terminal’s local storage and persistently stores it in the Terminal.
<b>LIFE CYCLE EVENT DESCRIPTION</b>	
<b>CUSTOMER BENEFIT:</b>	<ul style="list-style-type: none"> <li>• The Installation of a Widget Resource allows the use of a Widget after each boot-up of the Terminal without the need to seek for the Widget Resource in the local storage again.</li> <li>• The Installation from a local storage allows a user to transfer a Widget Resource to a Terminal via various means like: <ul style="list-style-type: none"> <li>• Memory card</li> <li>• Bluetooth</li> <li>• IrDA®</li> <li>• E-mail</li> <li>• MMS</li> </ul> </li> </ul>

<b>ACTOR(S):</b>	<ul style="list-style-type: none"> <li>• User</li> <li>• Widget Engine</li> <li>• Widget Manager</li> </ul>	
<b>PRECONDITIONS:</b>	<ul style="list-style-type: none"> <li>• The Widget Engine is installed on the Terminal</li> <li>• The Widget Manager is installed on the Terminal</li> <li>• The Widget Resource is stored in a local storage of the Terminal</li> </ul>	
<b>WORKFLOW (ACTIVITY SEQUENCE):</b>		
	<b>ACTOR INTENTION</b>	<b>SYSTEM RESPONSIBILITY</b>
	1. The user initiates the launch of the Widget Manager on the Terminal	2. The Widget Manager is loaded and executed on the Terminal
	3. The user selects a locally stored Widget Resource in the Widget Manager and requests installation	4. The Widget Manager starts the Installation procedure for the selected Widget Resource
		5. The Widget Manager loads the Widget Resource from the local storage
		6. The Widget Manager proves that the Widget Resource's zip archive and its file entries conform to the W3C Widget specification [2]
		7. The Widget Manager stores the Widget Resource in the Terminal's default Widget Resource repository
		8. The Widget Manager adds the Widget Resource to the Widget Resource List

		9. The Widget Manager stores the Widget Resource's Widget Resource Time Stamp
		10. The Widget Manager generates a prompt on the Terminal UI informing the user about the successful installation and asking to proceed with the Widget Processing
	Variant A	
	14. The user confirms to proceed with the Widget Processing	15. The Widget Manager asks the Widget Engine to process the Widget Resource
		16. The Widget Engine processes the Widget Resource
		17. The Widget Manager completes the Installation procedure
	Variant B	
	14. The user closes the prompt	15. The Widget Manager completes the Installation procedure
<b>POST CONDITIONS:</b>	<ul style="list-style-type: none"> <li>• The installed Widget Resource is persistently stored in the Terminal</li> <li>• The updated Widget List is persistently stored in the Terminal</li> <li>• The Widget Resource Time Stamp is persistently stored in the Terminal</li> </ul>	
<b>APPLICABILITY:</b>		
<b>EXCEPTIONS:</b>	None	



<b>REFERENCES:</b>	
--------------------	--

## ***2.2 WIDGET RESOURCE SETTINGS MANAGEMENT***

### ***2.2.1 AUTO-START AT TERMINAL BOOT-UP***

<b>USE CASE NAME</b>	Widget Settings Management for automatic Widget Resource processing after Terminal boot-up	
<b>SHORT DESCRIPTION:</b>	This use case describes the auto-start settings that can be managed for a Widget Resource	
<b>LIFE CYCLE EVENT DESCRIPTION</b>		
<b>CUSTOMER BENEFIT:</b>	The user can define those Widget Resources that shall be automatically processed after a Terminal boot-up without the need to manually launch them	
<b>ACTOR(S):</b>	<ul style="list-style-type: none"> <li>• User</li> <li>• Widget Manager</li> </ul>	
<b>PRECONDITIONS:</b>	<ul style="list-style-type: none"> <li>• The Widget Manager is installed on the Terminal</li> <li>• At least one Widget Resource is installed on the Terminal</li> <li>• The Widget Resource has a reference in the Widget Resource List</li> </ul>	
<b>WORKFLOW (ACTIVITY SEQUENCE):</b>		
	<b>ACTOR INTENTION</b>	<b>SYSTEM RESPONSIBILITY</b>
	1. The user initiates the launch of the Widget Manager on the Terminal	2. The Widget Manager is loaded and executed on the Terminal
		3. The Widget Manager provides the Widget Resource List on the Terminal UI to select one Widget Resource
	4. The user selects one Widget Resource on the Widget Resource List and requests to define the auto-start settings for this	5. The Widget Manager starts the auto-start settings management procedure for the chosen Widget Resource

		6. The Widget Manager generates a prompt on the Terminal UI asking the user to activate the auto-start setting for the Widget Resource so that the Widget Resource Processing is automatically initiated during the Terminal boot procedure
	7. The user activates the option for the auto-start setting and requests storage of this auto-start setting for the Widget Resource	8. The Widget Manager stores the auto-start setting for the Widget Resource
		9. The Widget Manager completes the auto-start settings management procedure
<b>POST CONDITIONS:</b>	<ul style="list-style-type: none"> <li>The auto-start setting for the Widget Resource is persistently stored in the Terminal</li> </ul>	
<b>APPLICABILITY:</b>	This use case is applicable to installed Widget Resources only	
<b>EXCEPTIONS:</b>		
<b>REFERENCES:</b>		

## 2.3 WIDGET RESOURCE DE-INSTALLATION

### 2.3.1 REMOTELY INITIATED WIDGET RESOURCE DE-INSTALLATION

<b>USE CASE NAME</b>	Widget Resource de-installation
<b>SHORT DESCRIPTION:</b>	This use case describes how a user de-installs a Widget Resource
<b>LIFE CYCLE EVENT DESCRIPTION</b>	
<b>CUSTOMER BENEFIT:</b>	<ul style="list-style-type: none"> <li>• The de-installation of a Widget Resource frees memory for other data and keeps the Widget Resource List manageable</li> <li>• An authorised party discovers that a Widget Resource has rogue features and decides that this Widget Resource should not be used any longer to prevent further damage. A user does not need to keep track of such a withdrawal.</li> </ul>
<b>ACTOR(S):</b>	<ul style="list-style-type: none"> <li>• Remote De-installation Service</li> <li>• User</li> <li>• Widget Engine</li> <li>• Widget Manager</li> </ul>
<b>PRECONDITIONS:</b>	<ul style="list-style-type: none"> <li>• The Remote De-installation Service is able to communicate with the Terminal</li> <li>• The Remote De-installation Service's de-installation request contains the Widget Resources</li> <li>• The Widget Engine is installed on the Terminal</li> <li>• The Widget Manager is installed on the Terminal</li> <li>• The Widget Resource, that shall be de-installed, is installed on the Terminal</li> <li>• The installed Widget Resource contains the Widget ID</li> <li>• The Widget Resource has a reference in the Widget Resource List</li> </ul>
<b>WORKFLOW (ACTIVITY SEQUENCE):</b>	

ACTOR INTENTION	SYSTEM RESPONSIBILITY
1. The Remote De-installation Service sends a de-installation request to the Terminal to initiate the de-installation of the Widget Resource with the Widget ID	2. The Terminal discovers that the received request is to be handled by the Widget Manager and asks the Widget Manager to handle the request
	3. The Widget Manager takes over the de-installation request and starts the server-push de-installation procedure
	4. The Widget Manager obtains the Widget ID from the de-installation request and proves that a Widget Resource with this Widget ID is installed on the Terminal
6. The user confirms the server-push de-installation	5. The Widget Manager generates a prompt on the Terminal UI asking the user whether he/she wants to proceed with the server-push de-installation of the Widget Resource or to cancel the server-push de-installation
	7. If the Widget Resource is currently processed by the Widget Engine, the Widget Manager will terminate all active processes of the processed Widget Resource
	8. The Widget Manager deletes the reference to the Widget Resource in the Widget Resource List

		9. The Widget Manager deletes the Widget Resource in the Terminal and all of the data that was generated and stored in the Terminal by this Widget Resource
	10. The user closes the prompt	10. The Widget Manager generates a prompt on the Terminal UI informing the user about the successful Widget Resource de-installation
		11. The Widget Manager completes the server-push de-installation procedure for this Widget Resource
<b>POST CONDITIONS:</b>	<ul style="list-style-type: none"> <li>All data in the Terminal that was related with the Widget Resource is deleted</li> </ul>	
<b>APPLICABILITY:</b>	This use case is applicable to installed Widget Resources only	
<b>EXCEPTIONS:</b>	None	
<b>REFERENCES:</b>		

**2.3.2 MANUAL WIDGET RESOURCE DE-INSTALLATION**

<b>USE CASE NAME</b>	Widget Resource de-installation	
<b>SHORT DESCRIPTION:</b>	This use case describes how a user de-installs a Widget Resource	
<b>LIFE CYCLE EVENT DESCRIPTION</b>		
<b>CUSTOMER BENEFIT:</b>	The de-installation of a Widget Resource frees memory for other data and keeps the Widget Resource List manageable	
<b>ACTOR(S):</b>	<ul style="list-style-type: none"> <li>• User</li> <li>• Widget Manager</li> </ul>	
<b>PRECONDITIONS:</b>	<ul style="list-style-type: none"> <li>• The Widget Manager is installed on the Terminal</li> <li>• At least one Widget Resource is installed on the Terminal</li> <li>• The Widget Resource has a reference in the Widget Resource List</li> </ul>	
<b>WORKFLOW (ACTIVITY SEQUENCE):</b>		
	<b>ACTOR INTENTION</b>	<b>SYSTEM RESPONSIBILITY</b>
	1. The user initiates the launch of the Widget Manager on the Terminal	2. The Widget Manager is loaded and executed on the Terminal
		3. The Widget Manager provides the Widget Resource List on the Terminal UI to select one Widget Resource
	4. The user selects one Widget Resource on the Widget Resource List and requests to de-install this	5. The Widget Manager starts the de-installation procedure for the chosen Widget Resource

		6. If the Widget Resource is currently processed by the Widget Engine, the Widget Manager will terminate all active processes of the processed Widget Resource
		7. The Widget Manager deletes the reference to the Widget Resource in the Widget Resource List
		8. The Widget Manager deletes the Widget Resource in the Terminal and all of its data that were generated and stored in the Terminal by this Widget Resource
	10. The user closes the prompt	9. The Widget Manager generates a prompt on the Terminal UI informing the user about the successful Widget Resource de-installation
		11. The Widget Manager completes the de-installation procedure for this Widget Resource
<b>POST CONDITIONS:</b>	<ul style="list-style-type: none"> <li>All data in the Terminal that were related with the Widget Resource are deleted</li> </ul>	
<b>APPLICABILITY:</b>	This use case is applicable to installed Widget Resources only	
<b>EXCEPTIONS:</b>	None	
<b>REFERENCES:</b>		

## **2.4 WIDGET PROCESSING**

### **2.4.1 BASIC WIDGET PROCESSING**

This is a functional building block that is used in other Widget Processing use cases.

The workflow is based on the “Steps for Processing a Widget Resource” defined in [2].

<b>USE CASE NAME</b>	Basic Processing of a Widget Resource	
<b>SHORT DESCRIPTION:</b>	This use case describes how a Widget Resource is processed	
<b>LIFE CYCLE EVENT DESCRIPTION</b>		
<b>CUSTOMER BENEFIT:</b>		
<b>ACTOR(S):</b>	<ul style="list-style-type: none"> <li>Widget Engine</li> </ul>	
<b>PRECONDITIONS:</b>	<ul style="list-style-type: none"> <li>The Widget Engine is installed on the Terminal</li> <li>The Widget Engine has loaded the Widget Resource</li> </ul>	
<b>WORKFLOW (ACTIVITY SEQUENCE):</b>	<b>ACTOR INTENTION</b>	<b>SYSTEM RESPONSIBILITY</b>
		1. The Widget Engine proves that the Widget Resource's zip archive and its file entries conform to the W3C Widget specification [2]
		2. The Widget Engine locates the Digital Signature
		3. The Widget Engine processes the Digital Signature
		4. The Widget Engine locates the Configuration Document
		5. The Widget Engine processes the Configuration Document
		6. The Widget Engine instantiates the Start File

		7. The Widget Engine may load additional files from the Widget Resource's zip archive and render them
		8. The Widget Engine may use other resources like local Terminal resources or remote resources external to the Terminal for rendering the Widget
<b>POST CONDITIONS:</b>	<ul style="list-style-type: none"> <li>The user is able to interact with the Widget</li> </ul>	
<b>APPLICABILITY:</b>	This use case is applicable to installed and non-installed Widget Resources	
<b>EXCEPTIONS:</b>	None	
<b>REFERENCES:</b>		

**2.4.2 WIDGET PROCESSING WITH NETWORK COVERAGE**

<b>USE CASE NAME</b>	Widget Processing with network coverage	
<b>SHORT DESCRIPTION:</b>	This use case describes the processing of a Widget on a Terminal that has network coverage and is able to communicate with and via the network	
<b>LIFE CYCLE EVENT DESCRIPTION</b>		
<b>CUSTOMER BENEFIT:</b>	<ul style="list-style-type: none"> <li>• A Widget is able to communicate with and via the network to retrieve latest data or configure a voice call forwarding service in a cellular network for instance.</li> <li>• The Widget Resource version update can also keep the Widget Resource up-to-date for bug fixing or feature enhancement for instance.</li> </ul>	
<b>ACTOR(S):</b>	<ul style="list-style-type: none"> <li>• Network</li> <li>• Widget</li> <li>• Widget Engine</li> </ul>	
<b>PRECONDITIONS:</b>	<ul style="list-style-type: none"> <li>• A Widget Resource is processed by the Widget Engine so that the Widget is rendered and potentially realised on the Terminal UI</li> </ul>	
<b>WORKFLOW (ACTIVITY SEQUENCE):</b>		
	<b>ACTOR INTENTION</b>	<b>SYSTEM RESPONSIBILITY</b>
	Variant A	
	1. The Widget starts a communication with the network by using resources of the Widget Engine	2. The network consumes the received data and may respond on them
Variant B		

	1. The network starts a communication with the Widget by using resources of the Widget Engine like triggering an event for which a Widget Resource is registered for	2. The Widget consumes the received data and may respond on them
<b>POST CONDITIONS:</b>		
<b>APPLICABILITY:</b>	This use case is applicable to installed and non-installed Widget Resources	
<b>EXCEPTIONS:</b>		
<b>REFERENCES:</b>		

**2.4.3 WIDGET PROCESSING WITHOUT NETWORK COVERAGE**

<b>USE CASE NAME</b>	Widget Processing without network coverage	
<b>SHORT DESCRIPTION:</b>	This use case describes the processing of a Widget on a Terminal that has <u>no</u> network coverage and is <u>not</u> able to communicate with and via the network	
<b>LIFE CYCLE EVENT DESCRIPTION</b>		
<b>CUSTOMER BENEFIT:</b>	<ul style="list-style-type: none"> <li>• A Widget is able to work with all local Terminal resources including but not limited to any data generated and locally stored in the Terminal by the Widget.</li> <li>• The use of Widgets without network coverage enables a user to interact with a Widget that does not require to permanently or even temporarily communicate with or via the network.</li> </ul>	
<b>ACTOR(S):</b>	<ul style="list-style-type: none"> <li>• Terminal</li> <li>• Widget</li> <li>• Widget Engine</li> </ul>	
<b>PRECONDITIONS:</b>	<ul style="list-style-type: none"> <li>• A Widget Resource is processed by the Widget Engine so that the Widget is rendered and potentially realised on the Terminal UI</li> </ul>	
<b>WORKFLOW (ACTIVITY SEQUENCE):</b>		
	<b>ACTOR INTENTION</b>	<b>SYSTEM RESPONSIBILITY</b>
	Variant A	
	1. The Widget requests to use a local Terminal resource by using resources of the Widget Engine	2. The Widget Engine fulfils the request and responds accordingly
Variant B		

	1. The Terminal generates an event that triggers the Widget Engine to process a Widget Resource	2. The Widget generates data and locally stores them
<b>POST CONDITIONS:</b>		
<b>APPLICABILITY:</b>	This use case is applicable to installed and non-installed Widget Resources	
<b>EXCEPTIONS:</b>	None	
<b>REFERENCES:</b>		

**2.4.4 PROCESSING OF MULTIPLE WIDGETS**

<b>USE CASE NAME</b>	Processing of multiple Widget Resources	
<b>SHORT DESCRIPTION:</b>	This use case describes the multi-tasking processing of multiple Widget Resources	
<b>LIFE CYCLE EVENT DESCRIPTION</b>		
<b>CUSTOMER BENEFIT:</b>	<ul style="list-style-type: none"> <li>• Widgets are rendered by the Widget Engine on the Terminal UI so that the illusion of parallelism is achieved. The user is able to choose one of these Widgets processed in parallel to interact with it without having the need to terminate one Widget Processing before he/she is able to interact with another Widget.</li> <li>• A Widget with which a user currently does not interact will be kept up-to-date as if it is processed by the Widget Engine as a single Widget.</li> </ul>	
<b>ACTOR(S):</b>	<ul style="list-style-type: none"> <li>• Terminal boot application</li> <li>• User</li> <li>• Widget Engine</li> <li>• Widget Resource</li> </ul>	
<b>PRECONDITIONS:</b>	<ul style="list-style-type: none"> <li>• The Widget Engine supports multi-tasking</li> </ul>	
<b>WORKFLOW (ACTIVITY SEQUENCE):</b>	The workflow below is given as an example for a wide range of use cases describing how the parallel processing of multiple Widget Resources is handled. It is the Terminal boot application in this use case that initiates the processing of Widget Resources. However, it could be the user or other Applications, too.	
	<b>ACTOR INTENTION</b>	<b>SYSTEM RESPONSIBILITY</b>
	1. The Terminal boot application initiates the processing of the Widget Resource A	2. The Widget Engine loads the Widget Resource A and starts the processing of the Widget Resource A as described in the use case “Basic Widget Processing” in section 2.4.1
	3. The Terminal boot application initiates the processing of the Widget	4. The Widget Engine loads the Widget Resource B and starts the processing of the

	Resource B	Widget Resource B as described in the use case “Basic Widget Processing” in section 2.4.1 without terminating the processing of the Widget Resource A
	5. The user toggles between the processed Widget A and Widget B on the Terminal UI to interact with one of them	6. The Widget Engine enables the user-selected Widget to interact with the user
<b>POST CONDITIONS:</b>	<ul style="list-style-type: none"> <li>The Terminal UI visualises which Widget is receptive for user input events</li> </ul>	
<b>APPLICABILITY:</b>	This use case is applicable to installed and non-installed Widget Resources	
<b>EXCEPTIONS:</b>	None	
<b>REFERENCES:</b>		

**2.4.5 WIDGET ENGINE EXECUTION**

The Widget Processing does not interfere with normal Terminal functionalities, e.g. the user can continue with usage of call, messaging or other Terminal Applications as before.

For example: where a Widget was active in the foreground and there is an incoming voice call, it shall automatically move to the background, unless the Widget has registered to deal with incoming calls.

The processing of Widgets shall also not interfere with any other Applications.

<b>USE CASE NAME</b>	Widget Engine Execution	
<b>SHORT DESCRIPTION:</b>	This use case describes how the Terminal manages the execution of the Widget Engine so that it does not hinder the execution of other Execution Environments or Applications	
<b>LIFE CYCLE EVENT DESCRIPTION</b>		
<b>CUSTOMER BENEFIT:</b>	<ul style="list-style-type: none"> <li>The user can use the Terminal in the same way as it is perceived by him/her without any Widget Resource processed by the Widget Engine</li> </ul>	
<b>ACTOR(S):</b>	<ul style="list-style-type: none"> <li>Execution Environment</li> <li>User</li> <li>SMS Application</li> <li>Widget Engine</li> </ul>	
<b>PRECONDITIONS:</b>	<ul style="list-style-type: none"> <li>The Execution Environment is installed in the Terminal</li> <li>The SMS Application is installed in the Terminal</li> <li>The Widget Engine is installed in the Terminal</li> <li>The Widget Engine is executed in the same Execution Environment like the SMS Application</li> </ul>	
<b>WORKFLOW (ACTIVITY SEQUENCE):</b>	The workflow below is given as an example for a wide range of use cases describing how a Widget Engine is executed in an Execution Environment in parallel to other installed Execution Environments and/or Applications	
	<b>ACTOR INTENTION</b>	<b>SYSTEM RESPONSIBILITY</b>
	1. The Widget Engine processes a Widget Resource which is realised on the Terminal UI and with which the user interacts	2. The SMS Application receives an SMS
		3. The SMS Application generates a prompt informing the user about the received SMS.

		4. The SMS Application asks the Execution Environment to display the prompt on the Terminal UI
		5. The Execution Environment displays the prompt on the Terminal UI so that the user can interact with it
	6. The user closes the prompt	7. The Execution Environment moves the Widget, that is processed by the Widget Engine, back into focus so that the user can interact with it again
<b>POST CONDITIONS:</b>		
<b>APPLICABILITY:</b>	This use case is applicable to installed and non-installed Widget Resources	
<b>EXCEPTIONS:</b>		
<b>REFERENCES:</b>		

**2.4.6 INSTALLED WIDGET PROCESSING**

**2.4.6.1 Start-up of installed Widget Resources during Terminal boot-up**

<b>USE CASE NAME</b>	Start-up of installed Widget Resources during Terminal boot-up	
<b>SHORT DESCRIPTION:</b>	This use case describes how installed Widget Resources are launched during the Terminal boot-up procedure	
<b>LIFE CYCLE EVENT DESCRIPTION</b>		
<b>CUSTOMER BENEFIT:</b>	<ul style="list-style-type: none"> <li>• A user can instantly use often used Widgets without the need to manually launch them after every Terminal boot-up</li> </ul>	
<b>ACTOR(S):</b>	<ul style="list-style-type: none"> <li>• Terminal boot application</li> <li>• User</li> <li>• Widget Engine</li> </ul>	
<b>PRECONDITIONS:</b>	<ul style="list-style-type: none"> <li>• At least one installed Widget Resource is configured to be processed at Terminal boot-up</li> <li>• The Widget Engine is installed on the Terminal</li> <li>• The Widget Engine supports multi-tasking</li> <li>• The Terminal boot application is installed on the Terminal</li> </ul>	
<b>WORKFLOW (ACTIVITY SEQUENCE):</b>	<b>ACTOR INTENTION</b>	<b>SYSTEM RESPONSIBILITY</b>
	1. The user switches the Terminal on	2. The Terminal boot application starts the Terminal boot procedure
		3. The Terminal boot application starts the auto-start procedure for applications that shall be launched during the Terminal boot procedure

		4. The Terminal boot application executes the list of applications to be launched
		5. The Terminal boot application asks the Widget Engine to process a Widget Resource per Widget Resource
		6. The Widget Engine processes each Widget Resource that it shall process as described in the use case “Basic Widget Processing” in section 2.4.1
		7. The Terminal boot application proceeds with any further steps of the auto-start procedure
		8. The Terminal boot application completes the auto-start procedure
		9. The Terminal boot application proceeds with any further steps of the Terminal boot procedure
		8. The Terminal boot application completes the Terminal boot procedure
<b>POST CONDITIONS:</b>	<ul style="list-style-type: none"> <li>Each launched Widget Resource is processed by the Widget Engine</li> </ul>	
<b>APPLICABILITY:</b>	This use case is applicable to installed Widget Resources only that are configured for the auto-start at Terminal boot-up (see section 2.2.1)	
<b>EXCEPTIONS:</b>	None	
<b>REFERENCES:</b>		

**2.4.6.2 Start-up of an installed Widget Resource by an application**

<b>USE CASE NAME</b>	Processing of an installed Widget Resource initiated by an Application	
<b>SHORT DESCRIPTION:</b>	This use case describes how an installed Widget Resource is launched by an Application that is executed in the Terminal	
<b>LIFE CYCLE EVENT DESCRIPTION</b>		
<b>CUSTOMER BENEFIT:</b>		
<b>ACTOR(S):</b>	<ul style="list-style-type: none"> <li>• Application</li> <li>• User</li> <li>• Widget Engine</li> </ul>	
<b>PRECONDITIONS:</b>	<ul style="list-style-type: none"> <li>• At least one Widget Resource is installed on the Terminal</li> <li>• The Widget Engine is installed on the Terminal</li> <li>• The Application is installed on the Terminal and is executed in an Execution Environment</li> </ul>	
<b>WORKFLOW (ACTIVITY SEQUENCE):</b>	<b>ACTOR INTENTION</b>	<b>SYSTEM RESPONSIBILITY</b>
	1. The Application asks the Widget Engine to process an installed Widget Resource and feeds the Widget Resource with data	2. The Widget Engine processes the Widget Resource as described in the use case “Basic Widget Processing” in section 2.4.1
		3. The Widget Resource processes the data fed to it by the Application
<b>POST CONDITIONS:</b>	<ul style="list-style-type: none"> <li>• The launched Widget Resource is processed by the Widget Engine</li> </ul>	
<b>APPLICABILITY:</b>	This use case is applicable to installed Widget Resources only	



<b>EXCEPTIONS:</b>	None
<b>REFERENCES:</b>	

**2.4.6.3 Manual start-up of an installed Widget**

<b>USE CASE NAME</b>	Manual processing of an installed Widget Resource	
<b>SHORT DESCRIPTION:</b>	The use case describes how a user manually launches an installed Widget Resource	
<b>LIFE CYCLE EVENT DESCRIPTION</b>		
<b>CUSTOMER BENEFIT:</b>	<ul style="list-style-type: none"> <li>The user decides which Widget Resource shall be processed by the Widget Engine so that it consumes only those Terminal resources, such as processing power or high-speed Execution memory, that are really necessary</li> </ul>	
<b>ACTOR(S):</b>	<ul style="list-style-type: none"> <li>User</li> <li>Widget Engine</li> <li>Widget Manager</li> </ul>	
<b>PRECONDITIONS:</b>	<ul style="list-style-type: none"> <li>The Widget Engine is installed on the Terminal</li> <li>The Widget Manager is installed on the Terminal</li> <li>At least one Widget Resource is installed in the Terminal</li> </ul>	
<b>WORKFLOW (ACTIVITY SEQUENCE):</b>		
	<b>ACTOR INTENTION</b>	<b>SYSTEM RESPONSIBILITY</b>
	1. The user initiates the launch of the Widget Manager on the Terminal	2. The Widget Manager is loaded and executed on the Terminal
		3. The Widget Manager provides the Widget Resource List on the Terminal UI to select one Widget Resource
	4. The user selects one Widget Resource on the Widget Resource List and requests to process this	5. The Widget Manager asks the Widget Engine to process the chosen Widget Resource

		6. The Widget Engine takes over the Widget Resource and starts the processing
		7. The Widget Engine processes the Widget Resource as described in the use case “Basic Widget Processing” in section 2.4.1
<b>POST CONDITIONS:</b>	<ul style="list-style-type: none"> <li>The user is able to interact with the Widget</li> </ul>	
<b>APPLICABILITY:</b>	This use case is applicable to installed Widget Resources only	
<b>EXCEPTIONS:</b>		
<b>REFERENCES:</b>		

**2.4.7 NON-INSTALLED WIDGET PROCESSING**

**2.4.7.1 Non-installed Widget Processing from a web server**

<b>USE CASE NAME</b>	Processing of a non-installed Widget Resource from a web server	
<b>SHORT DESCRIPTION:</b>	This use case describes how a Widget Resource is loaded from a web server and is processed without prior Installation	
<b>LIFE CYCLE EVENT DESCRIPTION</b>		
<b>CUSTOMER BENEFIT:</b>	<ul style="list-style-type: none"> <li>• A user does not want a Widget to persistently store any data on his Terminal. The on-the-fly processing prevents a Widget from locally storing data.</li> <li>• A user wants to use a Widget only once. Therefore, an installation for later use is not necessary.</li> </ul>	
<b>ACTOR(S):</b>	<ul style="list-style-type: none"> <li>• User</li> <li>• Widget Download Application</li> <li>• Widget Engine</li> <li>• Widget Manager</li> <li>• Widget Resource Provisioning Server</li> </ul>	
<b>PRECONDITIONS:</b>	<ul style="list-style-type: none"> <li>• The Widget Resource is provided for downloading on a Widget Resource Provisioning Server</li> <li>• The Terminal is able to communicate via TCP/IP with the Widget Resource Provisioning Server</li> <li>• The Widget Download Application is installed on the Terminal</li> <li>• The Widget Engine is installed on the Terminal</li> <li>• The Widget Manager is installed on the Terminal</li> </ul>	
<b>WORKFLOW (ACTIVITY SEQUENCE):</b>		
	<b>ACTOR INTENTION</b>	<b>SYSTEM RESPONSIBILITY</b>

	<p>1. The user initiates the launch of the Widget Download Application on the Terminal</p>	<p>2. The Widget Download Application is loaded and executed on the Terminal</p>
	<p>3. The user enters the URL of the Widget Resource on the Widget Resource Provisioning Server into the Widget Download Application and requests retrieval of this Widget Resource</p>	<p>4. The Widget Download Application retrieves the Widget Resource from the Widget Resource Provisioning Server</p>
		<p>5. The Widget Download Application discovers that the downloaded resource is a Widget Resource and asks the Terminal to handle the Widget Resource by the default application that is registered for handling Widget Resources</p>
		<p>6. The Widget Manager takes over the Widget Resource and starts the Installation procedure</p>
		<p>7. The Widget Manager generates a prompt on the Terminal UI asking the user to proceed with the Widget Resource Installation, to process the Widget Resource without installation or to cancel the Installation procedure</p>
	<p>8. The user confirms to proceed with the Widget Resource Processing without Installation</p>	<p>9. The Widget Manager asks the Widget Engine to process the Widget Resource</p>

		10. The Widget Engine takes over the Widget Resource and starts the processing
		11. The Widget Manager completes the Installation procedure
		12. The Widget Engine processes the Widget Resource as described in the use case “Basic Widget Processing” in section 2.4.1
<b>POST CONDITIONS:</b>	<ul style="list-style-type: none"> <li>The user is able to interact with the Widget</li> </ul>	
<b>APPLICABILITY:</b>	This use case is applicable to non-installed Widget Resources only	
<b>EXCEPTIONS:</b>	None	
<b>REFERENCES:</b>		

**2.4.7.2 Non-installed Widget Processing from a local storage**

<b>USE CASE NAME</b>	Processing of a non-installed Widget Resource from local storage	
<b>SHORT DESCRIPTION:</b>	This use case describes how a Widget Resource is loaded from local storage and processed without prior Installation	
<b>LIFE CYCLE EVENT DESCRIPTION</b>		
<b>CUSTOMER BENEFIT:</b>	<ul style="list-style-type: none"> <li>• A user does not want a Widget to persistently store any data on his Terminal. The on-the-fly processing prevents a Widget from locally storing data.</li> <li>• A user wants to use a Widget only once. Therefore, an installation for later use is not necessary.</li> <li>• The Installation from local storage allows a user to transfer a Widget Resource to a Terminal via various means like:                             <ul style="list-style-type: none"> <li>○ Memory card</li> <li>○ Bluetooth</li> <li>○ IrDA<sup>®</sup></li> <li>○ E-mail</li> <li>○ MMS</li> </ul> </li> </ul>	
<b>ACTOR(S):</b>	<ul style="list-style-type: none"> <li>• User</li> <li>• Widget Engine</li> <li>• Widget Manager</li> </ul>	
<b>PRECONDITIONS:</b>	<ul style="list-style-type: none"> <li>• The Widget Engine is installed on the Terminal</li> <li>• The Widget Manager is installed on the Terminal</li> <li>• The Widget Resource is stored in the Terminal</li> </ul>	
<b>WORKFLOW (ACTIVITY SEQUENCE):</b>		
	<b>ACTOR INTENTION</b>	<b>SYSTEM RESPONSIBILITY</b>
	1. The user initiates the launch of the Widget Manager on the Terminal	2. The Widget Manager is loaded and executed on the Terminal

	3. The user selects a locally stored Widget Resource in the Widget Manager and requests to process it	4. The Widget Manager starts the Installation procedure for the selected Widget Resource and loads the Widget Resource
		5. The Widget Manager generates a prompt on the Terminal UI asking the user to proceed with the Widget Resource Installation, to process the Widget without installation or to cancel the Installation procedure
	8. The user confirms to proceed with the Widget Processing without Installation	9. The Widget Manager asks the Widget Engine to process the Widget
		10. The Widget Engine takes over the Widget Resource and starts the processing
		11. The Widget Manager completes the Installation procedure
		12. The Widget Engine processes the Widget Resource as described in the use case "Basic Widget Processing" in section 2.4.1
<b>POST CONDITIONS:</b>	<ul style="list-style-type: none"> <li>The user is able to interact with the Widget</li> </ul>	
<b>APPLICABILITY:</b>	This use case is applicable to non-installed Widget Resources only	
<b>EXCEPTIONS:</b>	None	
<b>REFERENCES:</b>		

## 2.5 WIDGET PROCESSING TERMINATION

<b>USE CASE NAME</b>	Termination of a processed Widget Resource	
<b>SHORT DESCRIPTION:</b>	This use case describes how the processing of a Widget Resource is terminated	
<b>LIFE CYCLE EVENT DESCRIPTION</b>		
<b>CUSTOMER BENEFIT:</b>	<ul style="list-style-type: none"> <li>• A user is able to free high-speed Execution memory and processing power by terminating the processing of a Widget Resource that he/she no longer needs for instant access</li> </ul>	
<b>ACTOR(S):</b>	<ul style="list-style-type: none"> <li>• User</li> <li>• Widget Engine</li> <li>• Widget Manager</li> </ul>	
<b>PRECONDITIONS:</b>	<ul style="list-style-type: none"> <li>• The Widget Engine is installed on the Terminal</li> <li>• The Widget Manager is installed on the Terminal</li> <li>• At least one Widget Resource is processed by the Widget Engine</li> </ul>	
<b>WORKFLOW (ACTIVITY SEQUENCE):</b>		
	<b>ACTOR INTENTION</b>	<b>SYSTEM RESPONSIBILITY</b>
	1. The user initiates the launch of the Widget Manager on the Terminal	2. The Widget Manager is loaded and executed on the Terminal
		3. The Widget Manager provides the Widget Processing List on the Terminal UI to select one Widget Resource for processing termination
	4. The user selects one Widget Resource on the Widget Processing List and requests termination	5. The Widget Manager terminates the active process of the processed Widget Resource

<b>POST CONDITIONS:</b>	
<b>APPLICABILITY:</b>	
<b>EXCEPTIONS:</b>	None
<b>REFERENCES:</b>	

## 2.6 WIDGET RESOURCE VERSION UPDATE

This section covers use cases that describe automatically or manually initiated version updates managed by the Widget Manager and remotely triggered version updates. It does not cover a version update mechanism that is managed by a Widget.

The use cases reflect the W3C "Widgets 1.0: Updates" specification [3] regarding the use of an Update Description Document (UDD).

**2.6.1 INSTALLED WIDGET RESOURCE VERSION UPDATE**

**2.6.1.1 Automatic version update**

<b>USE CASE NAME</b>	Automatic version update of an installed Widget Resource
<b>SHORT DESCRIPTION:</b>	This use case describes how an installed Widget Resource is automatically updated.
<b>LIFE CYCLE EVENT DESCRIPTION</b>	
<b>CUSTOMER BENEFIT:</b>	<ul style="list-style-type: none"> <li>• A Widget Resource may have been updated on the Widget Provisioning Server after it has been downloaded and installed on the Terminal.</li> <li>• An update may have different reasons like bug fixing or feature enhancement.</li> <li>• The automatic Widget Resource version update shall prevent the use of outdated Widget Resources.</li> </ul>
<b>ACTOR(S):</b>	<ul style="list-style-type: none"> <li>• User</li> <li>• Widget Engine</li> <li>• Widget Manager</li> <li>• Widget Resource Provisioning Server</li> <li>• Widget Resource Update Server</li> </ul>

<p><b>PRECONDITIONS:</b></p>	<ul style="list-style-type: none"> <li>• The Terminal is able to communicate via TCP/IP with the Widget Resource Provisioning Server</li> <li>• The Terminal is able to communicate via TCP/IP with the Widget Resource Update Server</li> <li>• The Widget Engine is installed on the Terminal</li> <li>• The Widget Manager is installed on the Terminal</li> <li>• The installed Widget Resource contains a Widget Resource Update URL of a Widget Resource Update Server. This URL does not need to be the same URL where the installed Widget Resource was downloaded from</li> <li>• The Widget Resource Provisioning Server provides the latest version of the Widget Resource that shall be updated</li> <li>• The Widget Resource Update Server provides the Update Description Document (UDD) for the Widget Resource that shall be updated</li> </ul>	
<p><b>WORKFLOW (ACTIVITY SEQUENCE):</b></p>		
	<p><b>ACTOR INTENTION</b></p>	<p><b>SYSTEM RESPONSIBILITY</b></p>
	<p>1. The Terminal generates an event or the Widget Manager itself generates an event, that triggers the Widget Manager to start the automatic version update procedure for one of the installed Widget Resources</p>	
	<p>2. The Widget Manager obtains the Widget Resource Update URL for this Widget Resource</p>	

	<p>3. The Widget Manager sends a request to the Widget Resource Update Server to retrieve the UDD referenced by the Widget Resource Update URL in the Widget Resource</p>	<p>4. The Widget Resource Update Server responds with the requested UDD</p>
	<p>5. The Widget Manager compares the version in the UDD with the currently installed Widget Resource's version</p>	
	<p>Variant A: The Widget Resource's version in the UDD differs from the currently installed Widget Resource's version</p>	
	<p>6. The Widget Manager generates a prompt on the Terminal UI asking the user whether he/she wants to download and install an updated version of the Widget Resource or to cancel the automatic version update</p>	<p>7. The user confirms the download and installation</p>
	<p>8. The Widget Manager sends a request to the Widget Resource Provisioning Server to retrieve the Widget Resource referenced by the Widget Resource URL in the UDD</p>	<p>9. The Widget Resource Provisioning Server responds with the requested Widget Resource</p>
	<p>10. The Widget Manager replaces the installed Widget Resource by the retrieved Widget Resource</p>	

	11. The Widget Manager generates a prompt on the Terminal UI informing the user that the Widget Resource was successfully updated	12. The user closes the prompt
	13. The Widget Manager discovers that the Widget Resource is currently processed by the Widget Engine	
	14. The Widget Manager terminates the Widget Engine's processing of the Widget Resource	
	15. The Widget Manager asks the Widget Engine to process the Widget Resource	
	16. The Widget Engine processes the Widget Resource	
	17. The Widget Manager completes the version update procedure for the Widget Resource	
	Variant B: The Widget Resource's version in the UDD is the same as the currently installed Widget Resource's version	
6. The Widget Manager completes the version update procedure for this Widget Resource		
<b>POST CONDITIONS</b>	<ul style="list-style-type: none"> <li>• The replaced Widget Resource is persistently stored in the Terminal</li> <li>• The version update procedure does not modify any data that was generated and persistently stored by the Widget before the replacement</li> </ul>	



<b>APPLICABILITY:</b>	This use case is applicable to installed Widget Resources only
<b>EXCEPTIONS:</b>	None
<b>REFERENCES:</b>	

**2.6.1.2 Remotely initiated version update**

<b>USE CASE NAME</b>	Remotely initiated version update of an installed Widget Resource	
<b>SHORT DESCRIPTION:</b>	This use case describes how the version update of an installed Widget Resource is initiated by a remote service.	
<b>LIFE CYCLE EVENT DESCRIPTION</b>		
<b>CUSTOMER BENEFIT:</b>	<ul style="list-style-type: none"> <li>• A Widget Resource may have been updated on the Widget Provisioning Server after it has been downloaded and installed on the Terminal.</li> <li>• An update may have different reasons like bug fixing or feature enhancement.</li> <li>• The remotely triggered Widget Resource version update can prevent the use of outdated Widget Resources as soon as a new version is available.</li> </ul>	
<b>ACTOR(S):</b>	<ul style="list-style-type: none"> <li>• Remote Update Service</li> <li>• User</li> <li>• Widget Engine</li> <li>• Widget Manager</li> <li>• Widget Resource Provisioning Server</li> </ul>	
<b>PRECONDITIONS:</b>	<ul style="list-style-type: none"> <li>• The Terminal is able to communicate via TCP/IP with the Widget Resource Provisioning Server</li> <li>• The Remote Update Service is able to communicate with the Terminal</li> <li>• The Widget Engine is installed on the Terminal</li> <li>• The Widget Manager is installed on the Terminal</li> <li>• The Widget Manager is registered in the Terminal to handle push messages for Widget Resource version updates</li> </ul>	
<b>WORKFLOW (ACTIVITY SEQUENCE):</b>		
	<b>ACTOR INTENTION</b>	<b>SYSTEM RESPONSIBILITY</b>

	<p>1. The Remote Update Service sends a UDD to the Widget Manager to initiate the update of a specific Widget Resource which triggers the Widget Manager's version update procedure</p>	<p>2. The Widget Manager checks whether the Widget Resource, that is referenced in the UDD and shall be updated, is installed on the Terminal</p>
	<p>Variant A: The Widget Resource is installed on the Terminal</p>	
		<p>3. The Widget Manager compares the version in the UDD with the currently installed Widget Resource's version</p>
	<p>Variant A.1: The Widget Resource's version in the UDD differs from the currently installed Widget Resource's version</p>	
	<p>5. The user confirms the download and installation</p>	<p>4. The Widget Manager generates a prompt on the Terminal UI asking the user whether he/she wants to download and install an updated version of the Widget Resource or to cancel the version update</p>
	<p>7. The Widget Resource Provisioning Server responds with the requested Widget Resource</p>	<p>6. The Widget Manager sends a request to the Widget Resource Provisioning Server to retrieve the Widget Resource referenced by the Widget Resource URL in the UDD</p>
		<p>8. The Widget Manager replaces the installed Widget Resource with the retrieved Widget Resource</p>

	10. The user closes the prompt	9. The Widget Manager generates a prompt on the Terminal UI informing the user that the Widget Resource was successfully updated
		11. The Widget Manager discovers that the Widget Resource is currently processed by the Widget Engine
		12. The Widget Manager terminates the Widget Engine's processing of the Widget Resource
		13. The Widget Manager asks the Widget Engine to process the Widget Resource
		14. The Widget Engine processes the Widget Resource
		15. The Widget Manager completes the version update procedure for this Widget Resource
	Variant A.2: The Widget Resource's version in the UDD is the same as the currently installed Widget Resource's version	
		4. The Widget Manager completes the version update procedure for this Widget Resource
	Variant B: The Widget Resource is <u>not</u> installed on the Terminal	
		5. The Widget Manager completes the version update procedure

<b>POST CONDITIONS</b>	<ul style="list-style-type: none"> <li>• The replaced Widget Resource is persistently stored in the Terminal</li> <li>• The version update procedure does not modify any data that were generated and persistently stored by the Widget before the replacement</li> </ul>
<b>APPLICABILITY:</b>	This use case is applicable to installed Widget Resources only
<b>EXCEPTIONS:</b>	None
<b>REFERENCES:</b>	

**2.6.1.3 Manual version update**

<b>USE CASE NAME</b>	Manual version update of an installed Widget Resource
<b>SHORT DESCRIPTION:</b>	This use case describes how an installed Widget Resource is manually updated.
<b>LIFE CYCLE EVENT DESCRIPTION</b>	
<b>CUSTOMER BENEFIT:</b>	<ul style="list-style-type: none"> <li>• A Widget Resource may have been updated on the Widget Provisioning Server after it has been downloaded and installed on the Terminal.</li> <li>• An update may have different reasons like bug fixing or feature enhancement.</li> <li>• The manual Widget Resource version update shall prevent the use of outdated Widget Resources.</li> </ul>
<b>ACTOR(S):</b>	<ul style="list-style-type: none"> <li>• User</li> <li>• Widget Engine</li> <li>• Widget Manager</li> <li>• Widget Resource Provisioning Server</li> <li>• Widget Resource Update Server</li> </ul>

<b>PRECONDITIONS:</b>	<ul style="list-style-type: none"> <li>• The Terminal is able to communicate via TCP/IP with the Widget Resource Provisioning Server</li> <li>• The Terminal is able to communicate via TCP/IP with the Widget Resource Update Server</li> <li>• The Widget Engine is installed on the Terminal</li> <li>• The Widget Manager is installed on the Terminal</li> <li>• At least one Widget Resource is installed on the Terminal</li> <li>• The installed Widget Resources contain a Widget Resource Update URL of a Widget Resource Update Server. This URL does not need to be the same URL where the installed Widget Resource was downloaded from</li> <li>• The Widget Resource Provisioning Server provides the latest version of the Widget Resource that shall be updated</li> <li>• The Widget Resource Update Server provides the UDD for the Widget Resource that shall be updated</li> </ul>	
<b>WORKFLOW (ACTIVITY SEQUENCE):</b>		
	<b>ACTOR INTENTION</b>	<b>SYSTEM RESPONSIBILITY</b>
	1. The user initiates the launch of the Widget Manager on the Terminal	2. The Widget Manager is loaded and executed on the Terminal
	4. The user selects one Widget Resource on the Widget Resource List and requests to update this	3. The Widget Manager provides the Widget Resource List on the Terminal UI to select one Widget Resource for the manual version update
		5. The Widget Manager starts the version update procedure for the chosen Widget Resource

		6. The Widget Manager obtains the Widget Resource Update URL for the Widget Resource
	8. The Widget Resource Update Server responds with the requested UDD	7. The Widget Manager sends a request to the Widget Resource Update Server to retrieve the UDD that is referenced by the Widget Resource Update URL
		9. The Widget Manager compares the version in the UDD with the currently installed Widget Resource's version
Variant A: The Widget Resource's version in the UDD differs from the currently installed Widget Resource's version		
	11. The user confirms the download and installation	10. The Widget Manager generates a prompt on the Terminal UI asking the user whether he/she wants to download and install an updated version of the Widget Resource or to cancel the version update
	13. The Widget Resource Provisioning Server responds with the requested Widget Resource	12. The Widget Manager sends a request to the Widget Resource Provisioning Server to retrieve the Widget Resource referenced by the Widget Resource URL in the UDD
		14. The Widget Manager replaces the installed Widget Resource with the retrieved Widget Resource

		15. The Widget Manager discovers that the Widget Resource is currently processed by the Widget Engine
		16. The Widget Manager terminates the Widget Engine's processing of the Widget Resource
		17. The Widget Manager asks the Widget Engine to process the Widget Resource
		18. The Widget Engine processes the Widget Resource
		19. The Widget Manager completes the version update procedure for the Widget Resource
	Variant B: The Widget Resource's version in the UDD is the same as the currently installed Widget Resource's version	
	11. The user closes the prompt	10. The Widget Manager generates a prompt on the Terminal UI informing the user that a new version of the Widget Resource was not available
		12. The Widget Manager completes the version update procedure
<b>POST CONDITIONS:</b>	<ul style="list-style-type: none"> <li>• The replaced Widget Resource is persistently stored in the Terminal</li> <li>• The version update process does not modify any data that was generated and persistently stored by the Widget before the replacement</li> </ul>	

<b>APPLICABILITY:</b>	This use case is applicable to installed Widget Resources only
<b>EXCEPTIONS:</b>	None
<b>REFERENCES:</b>	

**2.6.2 NON-INSTALLED WIDGET RESOURCE VERSION UPDATE**

**2.6.2.1 Automatic version update**

<b>USE CASE NAME</b>	Automatic version update of a processed non-installed Widget Resource
<b>SHORT DESCRIPTION:</b>	This use case describes how a processed non-installed Widget Resource is automatically updated.
<b>LIFE CYCLE EVENT DESCRIPTION</b>	
<b>CUSTOMER BENEFIT:</b>	<ul style="list-style-type: none"> <li>• A Widget Resource may have been updated on the Widget Resource Provisioning Server after it has been downloaded to the Terminal.</li> <li>• An update may have different reasons like bug fixing or feature enhancement. The automatic Widget Resource version update shall prevent the use of outdated Widget Resources.</li> <li>• Due to the fact that a Terminal may be used without reboot over weeks, a processed non-installed Widget Resource may be outdated in the meanwhile.</li> </ul>
<b>ACTOR(s):</b>	<ul style="list-style-type: none"> <li>• User</li> <li>• Widget Engine</li> <li>• Widget Manager</li> <li>• Widget Resource Provisioning Server</li> <li>• Widget Resource Update Server</li> </ul>

<p><b>PRECONDITIONS:</b></p>	<ul style="list-style-type: none"> <li>• The Terminal is able to communicate via TCP/IP with the Widget Resource Provisioning Server</li> <li>• The Terminal is able to communicate via TCP/IP with the Widget Resource Update Server</li> <li>• The Widget Manager is installed on the Terminal</li> <li>• At least one non-installed Widget Resource is processed by the Widget Engine</li> <li>• The processed Widget Resource contains a Widget Resource Update URL of a Widget Resource Update Server. This URL does not need to be the same URL where the installed Widget Resource was downloaded from</li> <li>• The Widget Resource Provisioning Server provides the latest version of the Widget Resource that shall be updated</li> <li>• The Widget Resource Update Server provides the UDD for the Widget Resource that shall be updated</li> </ul>	
<p><b>WORKFLOW (ACTIVITY SEQUENCE):</b></p>		
	<p><b>ACTOR INTENTION</b></p>	<p><b>SYSTEM RESPONSIBILITY</b></p>
	<p>1. The Terminal generates an event or the Widget Manager itself generates an event, that triggers the Widget Manager to start the automatic version update procedure for one of the processed non-installed Widget Resources</p>	
	<p>2. The Widget Manager obtains the Widget Resource Update URL for the Widget Resource</p>	

	<p>3. The Widget Manager sends a request to the Widget Resource Update Server to retrieve the Update Description Document (UDD) referenced by the Widget Resource Update URL in the Widget Resource</p>	<p>4. The Widget Resource Update Server responds with the requested UDD</p>
	<p>5. The Widget Manager compares the version in the UDD with the currently installed Widget Resource's version</p>	
	<p>Variant A: The Widget Resource's version in the UDD differs from the currently installed Widget Resource's version</p>	
	<p>6. The Widget Manager generates a prompt on the Terminal UI asking the user whether he/she wants to download and process an updated version of the Widget Resource or to cancel the automatic version update</p>	<p>7. The user confirms the download and processing</p>
	<p>8. The Widget Manager sends a request to the Widget Resource Provisioning Server to retrieve the Widget Resource referenced by the Widget Resource URL in the UDD</p>	<p>9. The Widget Resource Provisioning Server responds with the requested Widget Resource</p>
	<p>10. The Widget Manager replaces the temporarily stored Widget Resource with the retrieved Widget Resource</p>	

	11. The Widget Manager terminates the Widget Engine's processing of the Widget Resource	
	12. The Widget Manager asks the Widget Engine to process the Widget Resource	
	13. The Widget Engine processes the Widget Resource	
	14. The Widget Manager completes the automatic version update procedure for the Widget Resource	
	Variant B: The Widget Resource's version in the UDD is the same as the currently processed Widget Resource's version	
	6. The Widget Manager completes the automatic version update procedure for this Widget Resource	
<b>POST CONDITIONS</b>	<ul style="list-style-type: none"> <li>The replaced Widget Resource is temporarily stored in the Terminal</li> </ul>	
<b>APPLICABILITY:</b>	This use case is applicable to processed non-installed Widget Resources only	
<b>EXCEPTIONS:</b>	None	
<b>REFERENCES:</b>		

**2.6.2.2 Manual version update**

<b>USE CASE NAME</b>	Manual version update of a processed non-installed Widget Resource
<b>SHORT DESCRIPTION:</b>	This use case describes how a processed non-installed Widget Resource is manually updated.
<b>LIFE CYCLE EVENT DESCRIPTION</b>	
<b>CUSTOMER BENEFIT:</b>	<ul style="list-style-type: none"> <li>• A Widget Resource may have been updated on the Widget Provisioning Server after it has been downloaded to the Terminal.</li> <li>• An update may have different reasons like bug fixing or feature enhancement. The automatic Widget Resource version update shall prevent the use of outdated Widget Resources.</li> <li>• Due to the fact that a Terminal may be used without reboot over weeks, a processed non-installed Widget Resource may be outdated in the meanwhile.</li> </ul>
<b>ACTOR(s):</b>	<ul style="list-style-type: none"> <li>• User</li> <li>• Widget Engine</li> <li>• Widget Manager</li> <li>• Widget Resource Provisioning Server</li> <li>• Widget Resource Update Server</li> </ul>

<b>PRECONDITIONS:</b>	<ul style="list-style-type: none"> <li>• The Terminal is able to communicate via TCP/IP with the Widget Resource Provisioning Server</li> <li>• The Terminal is able to communicate via TCP/IP with the Widget Resource Update Server</li> <li>• The Widget Engine is installed on the Terminal</li> <li>• The Widget Manager is installed on the Terminal</li> <li>• At least one non-installed Widget Resource is processed by the Widget Engine</li> <li>• The processed non-installed Widget Resource contains a Widget Resource Update URL of a Widget Resource Update Server. This URL does not need to be the same URL where the installed Widget Resource was downloaded from</li> <li>• The Widget Resource Provisioning Server provides the latest version of the Widget Resource that shall be updated</li> <li>• The Widget Resource Update Server provides an Update Description Document (UDD) for the Widget Resource that shall be updated</li> </ul>	
<b>WORKFLOW (ACTIVITY SEQUENCE):</b>		
	<b>ACTOR INTENTION</b>	<b>SYSTEM RESPONSIBILITY</b>
	1. The user initiates the launch of the Widget Manager on the Terminal	2. The Widget Manager is loaded and executed on the Terminal
	4. The user selects one Widget Resource on the Widget Processing List and requests to update this	3. The Widget Manager provides the Widget Processing List on the Terminal UI to select one Widget Resource for the manual version update
		5. The Widget Manager starts the version update procedure for the chosen Widget Resource

		6. The Widget Manager obtains the Widget Resource Update URL for the Widget Resource
	8. The Widget Resource Update Server responds with the requested UDD	7. The Widget Manager sends a request to the Widget Resource Update Server to retrieve the UDD that is referenced by the Widget Resource Update URL
		9. The Widget Manager compares the version in the UDD with the currently installed Widget Resource's version
Variant A: The Widget Resource's version in the UDD differs from the currently installed Widget Resource's version		
	11. The user confirms the download and processing	10. The Widget Manager generates a prompt on the Terminal UI asking the user whether he/she wants to download and process an updated version of the Widget Resource or to cancel the version update
	13. The Widget Resource Provisioning Server responds with the requested Widget Resource	12. The Widget Manager sends a request to the Widget Resource Provisioning Server to retrieve the Widget Resource referenced by the Widget Resource URL in the UDD
		14. The Widget Manager replaces the temporarily stored Widget Resource with the retrieved Widget Resource

		15. The Widget Manager terminates the Widget Engine's processing of the Widget Resource
		16. The Widget Manager asks the Widget Engine to process the Widget Resource
		17. The Widget Engine processes the Widget Resource
		18. The Widget Manager completes the version update procedure for the Widget Resource
	Variant B: The Widget Resource's version in the UDD is the same as the currently installed Widget Resource's version	
	11. The user closes the prompt	10. The Widget Manager generates a prompt on the Terminal UI informing the user that a new version of the Widget Resource was not available
		12. The Widget Manager completes the version update procedure
<b>POST CONDITIONS:</b>	<ul style="list-style-type: none"> <li>The replaced Widget Resource is temporarily stored in the Terminal</li> </ul>	
<b>APPLICABILITY:</b>	This use case is applicable to processed non-installed Widget Resources only	
<b>EXCEPTIONS:</b>	None	
<b>REFERENCES:</b>		

**2.6.3 AUTOMATIC VERSION UPDATE IN ROAMING MODE**

<b>USE CASE NAME</b>	Automatic Widget Resource version update in Roaming Mode	
<b>SHORT DESCRIPTION:</b>	This use case describes how a Widget Resource is <u>not</u> automatically updated when the Terminal is in the Roaming Mode.	
<b>LIFE CYCLE EVENT DESCRIPTION</b>		
<b>CUSTOMER BENEFIT:</b>	The automatic version update of a Widget Resource does not cause unexpected costs for a user when the Terminal is in the Roaming Mode.	
<b>ACTOR(S):</b>	<ul style="list-style-type: none"> <li>• Terminal boot application</li> <li>• User</li> <li>• Widget Manager</li> </ul>	
<b>PRECONDITIONS:</b>	<ul style="list-style-type: none"> <li>• The Terminal is attached to the home or a visited cellular network</li> <li>• The Widget Manager is installed on the Terminal</li> <li>• The automatic version update is deactivated for the Roaming Mode</li> </ul>	
<b>WORKFLOW (ACTIVITY SEQUENCE):</b>		
	<b>ACTOR INTENTION</b>	<b>SYSTEM RESPONSIBILITY</b>
	1. The user switches the Terminal on	2. The Terminal starts the Terminal boot application and loads and executes the Widget Manager during the boot-up procedure
		3. The Widget Manager loads its settings which contains the deactivation of the automatic version update in Roaming Mode.
Variant A: The Terminal attaches to a visited cellular network		

		4. The Terminal attaches to a visited cellular network
		5. The Widget Manager discovers that the Terminal is in the Roaming Mode. Therefore, the Widget Manager does not start the automatic version update procedure for installed and non-installed Widget Resources
	Variant A: The Terminal attaches to the home cellular network	
		4. The Terminal attaches to the home cellular network
		5. The Widget Manager discovers that the Terminal is <u>not</u> in the Roaming Mode. The Widget Manager starts the automatic version update procedure for installed and non-installed Widget Resources (see use cases in sections 2.6.1.1 and 2.6.2.1)
		6. The Terminal attaches to a visited cellular network
		7. The Widget Manager discovers that the Terminal is in the Roaming Mode.
		8. The Widget Manager completes the automatic version update procedure for installed and non-installed Widget Resources
<b>POST CONDITIONS:</b>		
<b>APPLICABILITY:</b>	This use case is applicable to installed and non-installed Widget Resources	

<b>EXCEPTIONS:</b>	None
<b>REFERENCES:</b>	<ul style="list-style-type: none"><li>• Use case “Automatic version update” in section 2.6.1.1</li><li>• Use case “Automatic version update” in section 2.6.2.1</li></ul>

## 2.7 WEB PAGE PROCESSING

From a functional architecture’s point of view, a Web Page is not processed by the Widget Engine but by the Web User Agent. A Web User Agent<sup>2</sup> may support a subset or a superset of the content formats that a Widget Engine supports. However, this does not necessarily mean that:

- a Web User Agent comprises a Widget Engine or
- a Widget Engine comprises a Web User Agent or
- a Widget Engine and a Web User Agent are implemented in the same software package.

---

<sup>2</sup> often called Browser

<b>USE CASE NAME</b>	Web Page Processing	
<b>SHORT DESCRIPTION:</b>	This use case generically describes how a Web User Agent retrieves a Web Page from a Web Page Provisioning Server and renders it	
<b>LIFE CYCLE EVENT DESCRIPTION</b>		
<b>CUSTOMER BENEFIT:</b>	<ul style="list-style-type: none"> <li>The user is able to consume content (i.e. Web Pages) that is provided on Web Page Provisioning Servers</li> </ul>	
<b>ACTOR(S):</b>	<ul style="list-style-type: none"> <li>User</li> <li>Web Page Provisioning Server</li> <li>Web User Agent</li> </ul>	
<b>PRECONDITIONS:</b>	<ul style="list-style-type: none"> <li>The Terminal is able to communicate via TCP/IP with the Web Page Provisioning Server</li> <li>A Web Page is provided for downloading on a Web Page Provisioning Server</li> <li>The Web User Agent is installed on the Terminal</li> </ul>	
<b>WORKFLOW (ACTIVITY SEQUENCE):</b>		
	<b>ACTOR INTENTION</b>	<b>SYSTEM RESPONSIBILITY</b>
	1. The user initiates the launch of the Web User Agent on the Terminal	2. The Web User Agent is loaded and executed on the Terminal
	3. The user enters the URL of the Web Page on the Web Page Provisioning Server into the Web User Agent and requests retrieval of this Web Page	4. The Web User Agent retrieves all components of the Web Page from the Web Page Provisioning Server
		5. The Web User Agent renders the Web Page
	6. The rendering result is displayed on the Terminal UI	

<b>POST CONDITIONS:</b>	<ul style="list-style-type: none"><li>• The user is able to interact with the Web Page</li></ul>
<b>APPLICABILITY:</b>	This use case is applicable to Web Pages only
<b>EXCEPTIONS:</b>	None
<b>REFERENCES:</b>	

### 3 DEFINITION OF TERMS

The table below contains the definition of terms used in this document.

TERM	DESCRIPTION
<b>APPLICATION</b>	<p>OMTP uses a broad definition of “Application” in this document.</p> <p>The term is used to cover active software components such as executables and libraries as well as more passive components such as content and scripts. The Application may be pre-loaded, downloaded to the mobile Terminal via means such as the mobile network, installed via another Application or transferred via infrared connection, Bluetooth, memory card or cable.</p> <p>Typical examples of mobile Applications include games, media players, word processors, security Applications and content.</p> <p>It does exclude firmware and SIM toolkit Applications.</p> <p>Depending on the Application Execution Environment, Applications may consist of one or more files with additional information such as the environment required to run the Application, debugging information, or other information used by the operating system to prepare the program to be run.</p>
<b>ACTIVE WIDGET</b>	Widget in full or minimized view, receptive for data updates
<b>BROWSER</b>	An Application that allows a user to access and view Web Pages. It is able to process and render Web Pages or any other content formats
<b>EXECUTION</b>	Occurs after Installation and Loading. The point in time begins when the Terminal starts the Application
<b>EXECUTION ENVIRONMENT</b>	A set of hardware and software components that provide facilities (e.g. Computing, Memory Management, input/output, etc.), necessary to support Applications
<b>HYBRID APPLICATION</b>	Applications that are authored using web formats that rely on a combination of locally and remotely hosted pages/files.

TERM	DESCRIPTION
<b>INSTALLATION</b>	The point when a package of software, that may contain one or more items of code and data, is integrated with the EE prior to Loading and Execution. Typically the software package will be resident in low speed memory, such as a file system or local database, after Installation
<b>LOADING</b>	The process of moving installed code into high-speed Execution memory. This could occur a long time prior to Execution. In some Terminals, this phase is just a matter of preparing the EE as the Execution memory is the same as the Installation memory
<b>MANUFACTURER</b>	The company that holds the warranty for the Terminals being sold in the market
<b>PLATFORM API</b>	Application Programming Interfaces (API) exposed by the native platform of the terminal
<b>REMOTE DE-INSTALLATION SERVICE</b>	The Remote De-installation Service initiates the Widget Resource de-installation of an installed Widget Resource
<b>REMOTE INSTALLATION SERVICE</b>	The Remote Installation Service initiates the Widget Resource Installation of a Widget Resource
<b>REMOTE UPDATE SERVICE</b>	The Remote Update Service initiates the Widget Resource version update of an installed Widget Resource
<b>ROAMING MODE</b>	The Terminal is attached to a visited cellular network which is not the home cellular network
<b>START FILE</b>	According to [2], the Start File is the first file that is instantiated after the configuration file was processed
<b>TERMINAL</b>	Used as an alternative term for a cellular telephone, device, handset, mobile equipment, user equipment or phone
<b>UPDATE DESCRIPTION DOCUMENT</b>	Contains information about a Widget Resource's latest release like version and size

TERM	DESCRIPTION
<b>WEB PAGE</b>	HTML or XHTML content that references further objects like Cascading Style Sheets (CSS), ECMAScript and graphics in various formats like PNG and GIF89
<b>WEB PAGE PROVISIONING SERVER</b>	Web server that provides a Web Page for downloading
<b>WEB RUNTIME ENVIRONMENT</b>	A software module that parses and executes browser formats (e.g. HTML and JavaScript), render web content to a display surface, and process user interaction (e.g., via keypad or touch screen) with that content. Installed widgets, web based applications, and hybrids will all typically use a WRE to render their content. Multiple Web Runtimes Environments may be installed on a device
<b>WEB USER AGENT</b>	see Browser
<b>WIDGET</b>	In [2], a Widget “is understood to be an interactive single purpose application for displaying and/or updating local data or data on the Web, packaged in a way to allow a single download and installation on a user’s machine, mobile phone, or mobile Internet device”
<b>WIDGET DOWNLOAD APPLICATION</b>	see Browser
<b>WIDGET ENGINE</b>	<p>A user agent that checks the conformance of Widget Resources or its configuration documents, extracts file entries from within a Widget Resource’s zip archive, renders various graphics formats, parses XML applications, manages a Document Object Model (DOM) and optionally supports HTTP, HTML, ECMAScript, CSS and other formats and specifications.</p> <p>The Widget Engine processes the Widget Resource</p>
<b>WIDGET ID</b>	According to [2], the Widget ID is a valid URI that specifies a unique identifier specific to the Widget Resource
<b>WIDGET LIST</b>	The Widget List contains a reference to each Widget installed on the Terminal. The user is able to choose a list item for rendering by the WRE

TERM	DESCRIPTION
<b>WIDGET MANAGER</b>	The Widget Manager manages the life cycle of a Widget Resource like Installation, De-installation and version update
<b>WIDGET PROCESSING</b>	<p>According to [2], the Widget Processing involves the following steps:</p> <ul style="list-style-type: none"> <li>• Acquire a Widget Resource Over HTTP or Local Storage</li> <li>• Verify the zip archive and its file entries</li> <li>• Locate the Digital Signature</li> <li>• Process the Digital Signature</li> <li>• Locate the Configuration Document</li> <li>• Process the configuration document</li> <li>• Instantiating the Start File</li> </ul>
<b>WIDGET PROCESSING LIST</b>	The Widget Processing List contains references to all Widget Resources that are currently processed by the Widget Engine
<b>WIDGET RESOURCE</b>	<p>[2] defines a Widget Resource as “a byte-stream or file that is a valid zip archive.                      A widget resource must contain a configuration document and may contain additional resources located either at the root of the archive or in sub-directories. In addition, a widget resource may also be digitally signed.”</p>
<b>WIDGET RESOURCE PROVISIONING SERVER</b>	Web server that provides a Widget Resource for downloading and is executed on the Terminal, on the SIM/UICC or a remote server
<b>WIDGET RESOURCE UPDATE SERVER</b>	Web server that provides a Widget Resource's Update Description Document for downloading
<b>WIDGET RESOURCE TIME STAMP</b>	The time stamp that is provided by the Widget Resource Provisioning Server in the HTTP response header
<b>WIDGET RESOURCE URL</b>	A Widget Resource's Uniform Resource Locator on the Widget Resource Provisioning Server

TERM	DESCRIPTION
<b>WIDGET RESOURCE UPDATE URL</b>	Uniform Resource Locator of a Widget Resource's Update Description Document on the Widget Resource Update Server which contains information about the Widget Resource's latest release version for instance.

### 3.1 MAPPING TO OTHER SDOs' TERMS

SDO	SDO's TERM	OMTP's TERM
W3C	Widget user agent	Widget Manager plus Widget Engine

## 4 ABBREVIATIONS

ABBREVIATION	DESCRIPTION
<b>AEE</b>	Application Execution Environment
<b>API</b>	Application Programming Interface
<b>DOM</b>	Document Object Model
<b>EE</b>	Execution Environment
<b>HTTP</b>	Hyper Text Transport Protocol
<b>HTTPS</b>	HTTP over SSL
<b>ME</b>	Mobile Equipment
<b>MMS</b>	Multimedia Messaging Service
<b>OMTP</b>	Open Mobile Terminal Platform
<b>OS</b>	Operating System
<b>UI</b>	User Interface
<b>SCWS</b>	Smartcard Web Server
<b>SDO</b>	Standards Development Organisation
<b>SMS</b>	Short Message Service
<b>SSL</b>	Secure Sockets Layer
<b>URI</b>	Uniform Resource Identifier
<b>UDD</b>	Update Description Document
<b>URL</b>	Uniform Resource Locator



ABBREVIATION	DESCRIPTION
<b>WRE</b>	Web Runtime Environment
<b>XML</b>	Extensible Markup Language

## 5 REFERENCED DOCUMENTS

No.	DOCUMENT	AUTHOR	DATE
1	RFC 2119 "Key words for use in RFCs to Indicate Requirement Levels". <a href="http://www.ietf.org/rfc/rfc2119.txt">http://www.ietf.org/rfc/rfc2119.txt</a>	S Bradner	March 1997
2	Widgets 1.0, W3C Working Draft 14 April 2008	W3C	April 2008
3	Widgets 1.0: Updates, W3C Working Draft 07 October 2008  <a href="http://www.w3.org/TR/2008/WD-widgets-updates-20081007/">http://www.w3.org/TR/2008/WD-widgets-updates-20081007/</a>	W3C	October 2008

----- END OF DOCUMENT -----