Gorba

imotion Update

Update Documentation

Technical Description

Version:	3.0
State:	Released
Classification:	Internal use only
Author:	RAN
Creation date:	2013-09-05
Repository:	$/Gorba/Main/Motion/Update/Documents/ TD_UpdateDocumentation.docx

Gorba AG
Sandackerstrasse
9245 Oberbüren
Switzerland

Table of contents

Modification management

Version	Date	Name	Dept.	Modifications	State
0.1	2013-09-05	RAN	SW	Initial version	draft
0.2	2013-09-12	RAN	SW	Added a new chapter 2, added sections to chapter 3, 4 and 5	draft
0.3	2013-09-20	WES	SW	Added pre- and post-installation actions Clarified update of config files	draft
0.4	2013-12-12	WES	SW	Added more information about FTP Improved information about update rollback and feedback Updated chapters about Medi Update	draft
1.1	2014-06-17	RAN	SW	Added chapter on parked update	draft
2.1	2014-10-23	RAN	SW	Refactor of chapter about configuration.	draft

Review

Version	Date	Name	Dept.	Remarks
0.4	2014-01-08	RAN	SW	Reviewed
1.1	2014-06-19	WES	SW	Reviewed: reordered chapters, added software version number
2.1	2014-11-10	WES	SW	Reviewed, added application icon

Release

Version	Date	Name	Dept.	Remarks
1.0	2014-01-08	WES	SW	First release matching Update 2.0.1402 (plus FTP)
2.0	2014-06-19	WES	SW	Release matching Update 2.2.1426.
3.0	2014-11-10	WES	SW	Release matching Update 2.4.1445.

1 Introduction

1.1 Update application version

This document covers Update version **2.4.1445**.

1.2 Scope

This document is a technical description of Update application. It gives all information to Gorba internal people to be able to work with this product.
This document is not intended to be a user manual.

The goal of this document is to describe in detail each Update parameter in order to configure it properly. It also provides the file types used in Update, the procedure for Update and the different configurations over which Update may be performed.
At the end of this document, the reader will have all the details to configure Update and use it.

1.3 Intended Audience

This document is addressed to product managers or customer project managers that are familiar with this product and are able to install, operate and maintain it.

2 Concept of update

Update application allows the user to update a TFT 2.0 system independently without any user interaction. A system is updated when it is running and does not require a shutdown. The user can update entire system, a single application, configuration or data through an update. The update.xml is used to configure the update application. The update application has three important sections which handle the processing of an update for a unit.

2.1 Update Provider

An update provider sends the update commands to a unit. It is also responsible for receiving the update feedback files and the log files from a unit. An update provider always communicates with an update client. Depending on the transfer technologies used for an update, different types of update providers are available in the update application.

2.2 Update Client

An update client receives the update commands from an update provider for one or more units. It also sends the update feedback files and log files of one or more units to the update provider. The update client uses the repository.xml. Depending on the transfer technologies used for an update, different types of update providers are available in the update application.

2.3 Update Agent

Each unit has just one update agent. The update agent is responsible for executing an update on the unit. The user can configure in update.xml to allow the unit to use the update agent on it for the execution of an update or only forward the update to other units but not update itself.

2.4 Transfer technologies used by update

Update application uses one or more of the following transfer technologies in order to get the required update and to send the feedback and log files of a unit after an update.

2.4.1 USB file transfer

Update of a unit can be done using a USB stick. In this case, the application generating the update uses an UsbUpdateProvider to send the update commands to a unit and receive the feedback and log files from the update client. Units that have to receive updates use an UsbUpdateClient which will download new updates as soon as a USB stick was detected.

2.4.2 Medi transfer

If a unit is connected to another unit via Medi, then a "master" unit can send the update over Medi to a "slave" unit. In this case, the update application on the "master" unit uses a MediUpdateProvider and the update application on the "slave" uses a MediUpdateClient. It is strongly suggested to use Medi transfer only in a stable local network – not over Wi-Fi or even GPRS/ UMTS.

2.4.3 FTP transfer

If a unit is connected to the internet (or an intranet, e.g. via VPN), then the icenter.Update can upload updates to an FTP server using the FtpUpdateProvider. The units then have to be configured to use an FtpUpdateClient to download updates from that FTP server.

3 Update.xml

The Update application requires configuration in order to operate correctly with the selected hardware. The file for configuration is **"Update.xml"**. Update also requires "**NLog.config**" for logging and **"medi.config"** for the Medi configuration.

3.1 Example Configuration

This chapter was automatically generated from Update.xml on 2014/11/10 by WES.

```
<?xml version="1.0" encoding="utf-8"?>
<Update xsi="http://www.w3.org/2001/XMLSchema-instance" schemaLocation="file:///. Update.xsd">
  <Agent Enabled="true">
    <!--     you can change the handled unit name manually by setting it here:
<UnitName>MyName</UnitName>     -->
    <RestartApplications>
      <Dependency Path="D:\Presentation\main.im2">
        <ExecutablePath>D:\Root\Progs\Composer\Composer.exe</ExecutablePath>
      </Dependency>
      <Dependency Path="D:\Presentation\Images">
        <ExecutablePath>D:\Root\Progs\Protran\Protran.exe</ExecutablePath>
        <ExecutablePath>D:\Root\Progs\HardwareManager\HardwareManager.exe</ExecutablePath>
      </Dependency>
    </RestartApplications>
    <ShowVisualization>true</ShowVisualization>
  </Agent>
  <Clients>
    <USBUpdateClient Name="USB_E">
      <ShowVisualization>true</ShowVisualization>
      <RepositoryBasePath>E:\Update</RepositoryBasePath>
      <USBDetectionTimeOut>PT20S</USBDetectionTimeOut>
      <PollInterval>PT30S</PollInterval>
    </USBUpdateClient>
    <USBUpdateClient Name="TestA">
      <ShowVisualization>true</ShowVisualization>
      <RepositoryBasePath>D:\Temp\UnitUpdate\TestA</RepositoryBasePath>
      <USBDetectionTimeOut>PT20S</USBDetectionTimeOut>
      <PollInterval>PT1H</PollInterval>
    </USBUpdateClient>
    <USBUpdateClient Name="TestB">
      <ShowVisualization>true</ShowVisualization>
      <RepositoryBasePath>D:\Temp\UnitUpdate\TestB</RepositoryBasePath>
      <USBDetectionTimeOut>PT20S</USBDetectionTimeOut>
      <PollInterval>PT1H</PollInterval>
    </USBUpdateClient>
    <FTPUpdateClient Name="FTP">
      <ShowVisualization>true</ShowVisualization>
      <Host>localhost</Host>
      <Port>21</Port>
      <Username>Gorba</Username>
      <Password>Gorba</Password>
      <RepositoryBasePath>/Gorba/Update</RepositoryBasePath>
      <PollInterval>PT10S</PollInterval>
    </FTPUpdateClient>
  </Clients>
  <Providers>
    <USBUpdateProvider Name="TestC">
      <RepositoryBasePath>D:\Temp\UnitUpdate\TestC</RepositoryBasePath>
    </USBUpdateProvider>
    <USBUpdateProvider Name="TestD">
      <ShowVisualization>false</ShowVisualization>
      <RepositoryBasePath>D:\Temp\UnitUpdate\TestD</RepositoryBasePath>
    </USBUpdateProvider>
    <USBUpdateProvider Name="TestE">
      <ShowVisualization>true</ShowVisualization>
      <RepositoryBasePath>A:\Temp\UnitUpdate\TestE</RepositoryBasePath>
    </USBUpdateProvider>
  </Providers>
  <Visualization HideTimeout="PT30S">
```

```
    <SplashScreen Enabled="true" X="0" Y="0" Width="1920" Height="630" />
    <LED Enabled="true" DefaultFrequency="1.25" ErrorFrequency="5" />
  </Visualization>
</Update>
```

Figure 1 Example Update.xml

3.2 XML Structure

This chapter was automatically generated from Update.xsd on 2014/11/10 by WES.
The root element is UpdateConfig.

3.2.1 UpdateConfig Structure

<Agent>	0:1	AgentConfig	All configuration parameters for the Update Agent.
<Clients>	0:1	List of <FTPUpdateClient> or <MediUpdateClient> or <USBUpdateClient>	List of all Update Clients that are available in Update.
<Providers>	0:1	List of <USBUpdateProvider> or <FTPUpdateProvider> or <MediUpdateProvider>	List of all Update Providers that are available in Update.
<Visualization>	0:1	VisualizationConfig	All configuration parameters for all visualization shown by Update like splash screen and LED.

Table 1 XML elements inside UpdateConfig

The Update configuration.

3.2.2 AgentConfig Structure

Enabled=	1:1	boolean	Flag indicating if the Update Agent is enabled. If the agent is disabled, this application will only forward updates for other applications, but not actually update the local system.

Table 2 XML attributes of AgentConfig

<UnitName>	0:1	string	The name of the Unit on which the Agent is present when it is other than the machine name. It allows overriding the use of the Machine Name as the local unit name.
<InstallationRoot>	0:1	string	The installation root directory name. It must be left empty.
<RestartApplications>	0:1	List of <Dependency>	List of all applications to be restarted for specific folder updates.
<ShowVisualization>	0:1	boolean	Flag indicating if the progress of an update is to be shown by Update visually with Splash screen or LED or both.

Table 3 XML elements inside AgentConfig

The Update Agent handles the actual process of updating a Unit. The configuration specifies if the Update Agent which must update the local system or just forward the received update to other Units and not update the local system. It also specified the restart of applications upon completion of an update based on the configuration. In certain situations, it might be required to run multiple Update

applications on one system (e.g. to simulate the update of an entire fleet). For this purpose, the "Agent" tag contains two additional sub-tags. These sub-tags should never be used on a productive system.

3.2.3 DependencyConfig Structure

Path=	1:1	string	Path of the folder or file whose update triggers the restart of the applications present at the executable paths.

Table 4 XML attributes of DependencyConfig

<ExecutablePath>	1:*	string	List of executable paths of the applications to be restarted if an update of folder or file present in the attribute Path occurs.

Table 5 XML elements inside DependencyConfig

The configuration of the dependancy of the update of a folder or file specified in the "Path" attribute which triggers the restart of the application specified in the "ExecutablePath". For a single dependancy, more than one application can be specified for restart by using multiple "ExecutablePath" configuration.

3.2.4 FtpUpdateClientConfig Structure

Name=	1:1	string	The unique name of the update client. It must contain only alpha-numeric characters. It is case-insensitive. It is a compulsory to define the name. If the name changes, all cached data for the client will be lost.

Table 6 XML attributes of FtpUpdateClientConfig

<ShowVisualization>	0:1	boolean	Flag indicating if the progress of an update is to be shown by Update visually with Splash screen or LED or both.
<Host>	1:1	string	The DNS host name or IP address of the FTP server used for FTP updates.
<Port>	0:1	int, default: "21"	The value of the FTP server TCP port.
<Username>	1:1	string	The FTP server login username.
<Password>	1:1	string	The FTP server login password. Mandatory, even if anonymous login is used.
<RepositoryBasePath>	0:1	string, default: ""	The source path of the update (on FTP server) to be installed.
<PollInterval>	0:1	duration, default: "PT5M"	The interval (expressed as XML duration) at which the repository of update is scanned. The admitted values are positive non-zero durations.

Table 7 XML elements inside FtpUpdateClientConfig

A FTPUpdateClient must be configured for each of the FTP servers from which the Unit can receive an update.

3.2.5 MediUpdateClientConfig Structure

Name=	1:1	string	The unique name of the update client. It must contain only alpha-numeric characters. It is case-insensitive. It is a compulsory to define the name. If the name changes, all cached data for the client will be lost.

Table 8 XML attributes of MediUpdateClientConfig

<ShowVisualization>	0:1	boolean	Flag indicating if the progress of an update is to be shown by Update visually with Splash screen or LED or both.

Table 9 XML elements inside MediUpdateClientConfig

A MediUpdateClient must be configured if the Unit can receive an update from another Unit. This is usually required in a Master-Slave environment. The slave Unit must have a MediUpdateClient configured if it has to receive updates from the master Unit.

3.2.6 UsbUpdateClientConfig Structure

Name=	1:1	string	The unique name of the update client. It must contain only alpha-numeric characters. It is case-insensitive. It is a compulsory to define the name. If the name changes, all cached data for the client will be lost.

Table 10 XML attributes of UsbUpdateClientConfig

<ShowVisualization>	0:1	boolean	Flag indicating if the progress of an update is to be shown by Update visually with Splash screen or LED or both.
<RepositoryBasePath>	1:1	string	The source path of the update (on the USB stick) to be installed.
<USBDetectionTimeOut>	0:1	duration, default: "PT20S"	The time to wait (expressed as XML duration) for the detection of the USB stick to ensure that it is definitely present before starting the update process.
<PollInterval>	0:1	duration	The interval (expressed as XML duration) at which the repository of update is scanned. The admitted values are positive non-zero durations. The poll interval value must be higher than the USBDetectionTimeOut value.

Table 11 XML elements inside UsbUpdateClientConfig

A UsbUpdateClient must be configured if the Unit has to receive updates via the USB stick. The Update command for a Unit on USB stick is deleted once an update is performed.

3.2.7 UsbUpdateProviderConfig Structure

Name=	0:1	string	The unique name of the Update Provider. It must contain only alpha-numeric characters. It is case-insensitive. It is a compulsory to define the name. If the name changes, all cached data for the Provider will be lost.

Table 12 XML attributes of UsbUpdateProviderConfig

<ShowVisualization>	0:1	boolean	Flag indicating if the progress of an update is to be shown by Update visually with Splash screen or LED or both.
<RepositoryBasePath>	1:1	string	The repository base path in which the repository.xml file can be found.

Table 13 XML elements inside UsbUpdateProviderConfig

A UsbUpdateProvider must be configured if the Unit has to send updates to a USB stick. It is not configured on a Unit.

3.2.8 FtpUpdateProviderConfig Structure

Name=	0:1	string	The unique name of the Update Provider. It must contain only alpha-numeric characters. It is case-insensitive. It is a compulsory to define the name. If the name changes, all cached data for the Provider will be lost.

Table 14 XML attributes of FtpUpdateProviderConfig

<ShowVisualization>	0:1	boolean	Flag indicating if the progress of an update is to be shown by Update visually with Splash screen or LED or both.
<Host>	1:1	string	The host name of the FTP server.
<Port>	0:1	int, default: "21"	The value of the FTP server TCP port.
<Username>	1:1	string	The FTP server login username.
<Password>	1:1	string	The FTP server login password.
<RepositoryBasePath>	0:1	string, default: ""	The source path of the update to be installed.
<PollInterval>	0:1	duration, default: "PT5M"	The interval at which the repository is scanned.
<Compression>	0:1	CompressionAlgorithm, default: "None"	The compression mode to be used by the FTP provider.

Table 15 XML elements inside FtpUpdateProviderConfig

A FtpUpdateProvider must be configured if the Unit has to send updates to an FTP server. It is not configured on a Unit.

3.2.9 CompressionAlgorithm Enumeration

None	No compression is applied on the files.
GZIP	The files are compressed using the GZIP algorithm (see RFC 1952).

Table 16 Enumeration values of CompressionAlgorithm

Possible compression algorithms.

3.2.10 MediUpdateProviderConfig Structure

Name=	0:1	string	The unique name of the Update Provider. It must contain only alpha-numeric characters. It is case-insensitive. It is a compulsory to define the name. If the name changes, all cached data for the Provider will be lost.

Table 17 XML attributes of MediUpdateProviderConfig

<ShowVisualization>	0:1	boolean	Flag indicating if the progress of an update is to be shown by Update visually with Splash screen or LED or both.

Table 18 XML elements inside MediUpdateProviderConfig

A MediUpdateProvider must be configured if the Unit has to send updates to other Units. This configuration is required on a Master Unit in a Master-Slave setup of Units.

3.2.11 VisualizationConfig Structure

HideTimeout=	0:1	duration, default: "PT30S"	The time after which any of the visualizations are hidden upon completion of the update progress.

Table 19 XML attributes of VisualizationConfig

<SplashScreen>	0:1	SplashScreenVisualization-Config	Configuration of the splash screen that can be shown by Update.
<LED>	0:1	LedVisualizationConfig	Configuration of the Update LED on InfoVision PC-2 and Infovision Compact hardware.

Table 20 XML elements inside VisualizationConfig

The configuration for the Update progress visualization. There are two options to visualize the progress of an update. The Splash screen and the Update LED. If the Unit has a Physical screen, the Splash screen visualization can be enabled. The Update LED is available on the InfoVision PC-2 and InfoVision Compact hardware.

3.2.12 SplashScreenVisualizationConfig Structure

Enabled=	1:1	boolean	Flag indicating if the splash screen is to be shown by Update.
X=	0:1	int	The horizontal location in pixels of the splash screen on the screen. If this attribute is not defined, the splash screen is shown at the very right of the screen.
Y=	0:1	int	The vertical location in pixels of the splash screen on the screen. If this attribute is not defined, the splash screen is shown at the very top of the screen.
Width=	0:1	int	The width of the splash screen in pixels. If this attribute is not defined, the splash screen is shown across the entire width of the screen.
Height=	0:1	int	The height of the splash screen in pixels. If this attribute is not defined, the splash screen is shown across the entire height of the screen.

Table 21 XML attributes of SplashScreenVisualizationConfig

The splash screen of the update progress can be configured to be shown or not. The location of the splash screen on the TFT can be configured so that it fits any type of display like widescreen etc.

3.2.13 LedVisualizationConfig Structure

Enabled=	1:1	boolean	Flag indicating if the LED is to be switched to indicate the update status by Update.
DefaultFrequency=	0:1	double, default: "1.25"	The default blink frequency in Hz at which the LED is switched to indicate transfer of update. Default value is 1.25 Hz means a change every 400 ms.
ErrorFrequency=	0:1	double, default: "5"	The error frequency in Hz at which the LED is switched to indicate update errors. Default value is 5 Hz means a change every 100 ms.

Table 22 XML attributes of LedVisualizationConfig

The Update LED visualization can be configured to be available or not. The Update LED gives a constant indication of the status of the update. When files are being transferred, the Update LED blinks at a specific frequency (configurable). When an upadte is being installed, the Update LED is on steadily. If an error occurs during update, the Update LED blinks at a specific frequency (configurable). it is recommeded to configure the transfer and the error frequency of the LED blinking differently. • Normal blinking – Indicates that the update is reading from/writing to the USB stick. • Slow blinking – Indicates that there was an error during an update. • Steady on – Indicates that an update is being processed. • Steady off – Indicates that the update is complete.

4 File types in Update

This section describes the different types of files used by the update application and their usage.

4.1 Repository.xml

The repository.xml is a XML file which describes the repository structure provided by an update provider and to be used by the update client to perform an update. You should never have to create nor edit a "repository.xml" file; this chapter serves only as a reference for the given structure.

```
<?xml version="1.0"?>
<Repository xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Config ValidFrom="1.0">
    <ResourceDirectory>Resources</ResourceDirectory>
    <CommandsDirectory>Commands</CommandsDirectory>
    <FeedbackDirectory>Feedback</FeedbackDirectory>
  </Config>
</Repository>
```

Figure 4-1 A repository.xml example

4.1.1 <Config>

This tag specifies the repository configuration valid for a version of update. Multiple <Config> tags are allowed. Hence, different configuration valid for different versions of update may be specified. This can be used to allow newer update software to install files from different folders in the repository. The first version configuration that matches the current version of update will be taken when multiple configurations are available.

Tag name	Sub-tags allowed	Attributes allowed
Config	• ResourceDirectory • CommandsDirectory • FeedbackDirectory	ValidFrom the version of update from which the config is valid. If the value is not set, the configuration is valid for all versions.

Table 4-1 Config content

4.1.1.1 <Config><ResourceDirectory>

This tag specifies the name of the sub-directory as a relative path which contains all the resources required for an update.

4.1.1.2 <Config><CommandsDirectory>

This tag specifies the name of the directory which contains sub-directories named after each unit which is to be updated. Each sub-directory named after a unit, contains all the commands for that specific unit which is required to perform an update.

4.1.1.3 <Config><FeedbackDirectory>

This tag specifies the name of the directory which contains sub-directories named after each unit from which feedback has been received. Each sub-directory named after a unit, contains all the feedback, including the feedback files and all the log files from the specific unit.

4.2 Command File

The command file contains details of the update for a specific unit. It contains the complete folder and file structure expected on the unit upon completion of the update. The extension for a command file is ".guc". Below is an example of the command file created for an update of a unit.

```
<?xml version="1.0"?>
<UpdateCommand xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" Version="2.0">
  <UpdateId BackgroundSystemGuid="338ce77c-a650-487e-a785-
6481f58fff09" UpdateIndex="2" />
  <UnitId Name="Unit1" />
  <ActivateTime>2013-08-19T00:00:00Z</ActivateTime>
  <PreInstallation>
    <Executable Name="Setup.exe" Hash="9BE11C0C7D71C1278ADA1A29DB442D16" Args="/s" />
    <File Name="setup.msi" Hash="478D1FAC08CD7101115D77E10FBF8D6D" />
  </PreInstallation>
  <Folder Name="Config">
    <Folder Name="Protran">
      <File Name="codeconversion.csv" Hash="B5FEF210D52CFF30930E78E685A1C4E9" />
      <File Name="dictionary.xml" Hash="AAD394EFB5E5F0E485F9BA6D4F343EE5" />
      <File Name="ibis.xml" Hash="740E311D87CB41BBD9402956F24C39BC" />
      <File Name="io.xml" Hash="4FCE3CC9743B70DE5E7695CF172DBE93" />
      <File Name="medi.config" Hash="7C6B028A54DB67B45CFBE80A249FC73F" />
      <File Name="NLog.config" Hash="056BC7E9A10BCD5F235855AB82027FE7" />
      <File Name="protran.xml" Hash="6AC44B364C5BAD62FFF30C45D9111C8C" />
      <File Name="specialtext.csv" Hash="A7E4E2EF019AC67E0E5F538B448CD662" />
    </Folder>
  </Folder>
  <PostInstallation>
    <Run Name="c:\windows\System32\shutdown.exe" Args="/r /t 0" />
  </PostInstallation>
</UpdateCommand>
```

Figure 4-2 Command file example

4.2.1 <UpdateId>

This tag specifies the identification for the specific update.

Tag name	Sub-tags allowed	Attributes allowed
UpdateId	• None	BackgroundSystemGuid specifies the id of the background system that created this command. UpdateIndex specifies the index for an update which is incremented for each update and is unique to a background system.

Table 4-2 UpdateId content

4.2.2 <UnitId>

This tag specifies the identification for a unit to which the update is destined.

Tag name	Sub-tags allowed	Attributes allowed
UnitId	• None	Name specifies name of the unit.

Table 4-3 UnitId content

4.2.3 <ActivateTime>

This tag specifies the UTC time at which the update must be activation on the unit.

4.2.4 <Folder>

This tag specifies the folder to be updated on the unit.

Tag name	Sub-tags allowed	Attributes allowed
Folder	• Folder • File	Name specifies name of the folder.

Table 4-4 Folder content

4.2.5 <File>

This tag specifies a file to be updated within a folder on the unit.

Tag name	Sub-tags allowed	Attributes allowed
File	• None	Name specifies name of the file. Hash is the MD5 hash of the file.

Table 4-5 File content

4.2.6 <PreInstallation> and <PostInstallation>

This tag specifies an action to be performed before or after the installation of the update.

Tag name	Sub-tags allowed	Attributes allowed
PreInstallation PostInstallation	• Executable • File • Folder • Run	none

Table 4-6 PreInstallation and PostInstallation content

4.2.7 <Executable>

This tag specifies an executable to be copied to a temporary folder on the unit and then executed.

Tag name	Sub-tags allowed	Attributes allowed
Executable	• None	Name specifies name of the executable. Hash is the MD5 hash of the executable. Args are the command line arguments to be provided to the executable when run.

Table 4-7 Executable content

4.2.8 <Run>

This tag specifies an executable to be run on the unit.

Tag name	Sub-tags allowed	Attributes allowed
Run	• None	Name specifies full path to the executable. Args are the command line arguments to be provided to the executable when run.

Table 4-8 Run content

4.3 Resource file

A resource file is a file to be updated on the unit. It has the extension ".rx" and the name of the file is the MD5 hash of the file with the extension.

4.4 Feedback files

There are two kinds of feedback files available from a unit. One is the feedback about the update which is a file with the extension ".guf". The second type of feedback files are logs files with the extension ".log".
Each update state has a ".guf" file for it. Please refer to chapter 5.5 for more information.

The name of the feedback file for the update state is of the following format.
"<Background System GUID>-<Update Index>-<Update state as integer>-<Update state>.guf".
Below is a description of the contents of some examples of feedback files.

4.4.1 Update State: Transferred Feedback file

```
<?xml version="1.0"?>
<UpdateStateInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" Version="2.0">
  <UpdateId BackgroundSystemGuid="c2c04030-8218-44a1-8df1-
ef2aaf16559e" UpdateIndex="16" />
  <UnitId TenantId="0" Name="PM800-215" />
  <TimeStamp>2013-09-12T10:04:33.590776Z</TimeStamp>
  <State>Transferred</State>
</UpdateStateInfo>
```

Figure 4-3 A feedback file for the update state transferred example

The XML tags in the file provide information regarding the update, unit and the update state. The tags `<UpdateId>` and `<UnitId>`, have been previously described in section 4.2.

4.4.1.1 `<TimeStamp>`

This tag specifies the time at which this feedback was created.

Tag name	Sub-tags allowed	Attributes allowed
TimeStamp	• None	None

Table 4-9 TimeStamp content

4.4.1.2 `<State>`

This tag specifies the update state.

Tag name	Sub-tags allowed	Attributes allowed
State	• None	None

Table 4-10 State content

4.4.2 Update State: Installed Feedback file

The feedback file for the update state Installed contains the complete file system structure after the update with a state for each file and folder apart from the Update state for the complete update.

```
<?xml version="1.0"?>
<UpdateStateInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" Version="2.0">
  <UpdateId BackgroundSystemGuid="c2c04030-8218-44a1-8df1-
ef2aaf16559e" UpdateIndex="16" />
  <UnitId TenantId="0" Name="PM800-215" />
  <TimeStamp>2013-09-12T10:04:42.70388Z</TimeStamp>
  <State>Installed</State>
  <Folder Name="Config" State="UpToDate">
    <Folder Name="Protran" State="UpToDate">
      <Folder Name="Recordings" State="UpToDate">
        <File Name="recording.PRO.CSV" State="UpToDate"
              Hash="7F4BC0BF695CBE9650DA7571362A8B1C"
              ExpectedHash="7F4BC0BF695CBE9650DA7571362A8B1C" />
      </Folder>
      <File Name="codeconversion.csv" State="UpToDate"
            Hash="B5FEF210D52CFF30930E78E685A1C4E9"
            ExpectedHash="B5FEF210D52CFF30930E78E685A1C4E9" />
      <File Name="dictionary.xml" State="UpToDate"
            Hash="AAD394EFB5E5F0E485F9BA6D4F343EE5"
            ExpectedHash="AAD394EFB5E5F0E485F9BA6D4F343EE5" />
    </Folder>
  </Folder>
</UpdateStateInfo>
```

Table 4-11 A feedback file for the update state installed example

4.4.2.1 <Folder>

This tag specifies the folder which was updated on the unit.

Tag name	Sub-tags allowed	Attributes allowed
Folder	• Folder • File	Name specifies name of the folder. State specifies the current update state of the folder.

Table 4-12 Folder content

4.4.2.2 <File>

This tag specifies a file which was updated within a folder on the unit.

Tag name	Sub-tags allowed	Attributes allowed
File	• None	Name specifies name of the file. State specifies the current update state of the file. Hash is the MD5 hash of the file. ExpectedHash is the expected MD5 hash of the resource that should have been copied to the given file name.

Table 4-13 File content

5 Update Procedure

This section describes the update procedure for a TFT 2.0 system using USB, FTP and Medi. Create an update required for the system on a USB using the process described in the document "TD_USBUpdateManager_UserManual".

5.1 System requirements to run an update

A TFT 2.0 system to be updated must satisfy the following conditions before an update can be performed on it. The system must be on and have the following application running on it:
SystemManager.exe
Update.exe

5.2 Update process

5.2.1 Receiving update via USB

Upon insertion of the USB stick which contains an update for the specific TFT 2.0 system (unit), the Update application detects the USB stick. Upon detection of the USB stick, update application waits until the configured timeout is fulfilled before starting the update process.

5.2.1.1 Downloads commands and resources for a unit from USB stick

The update process checks the USB stick for a Command which is valid for it and then verifies if all the resources required for the update specified in the Command are available. The Commands and the resources for the unit(s) are downloaded from the USB stick. After a successful download, the Commands are removed from the stick.

5.2.2 Receiving update via Medi

A slave unit is connected to the master unit and can receive its updates from the master unit. The Medi Update Client on the slave unit sends via Medi an Update Registration to the Medi Update Provider on the master unit. The Medi Update Provider on the master unit sends an Update Registration Acknowledgement to the slave unit.

5.2.2.1 Downloads commands and resources for a unit via Medi

The Medi Update Provider on the master unit registers to the available local Update clients like USB or FTP Update Client to receive the update commands and recourses. Once the Medi Update Provider on the master receives the commands, it sends all the resources for the update command via Medi to the registered units. It also sends the command for the unit to the unit's Medi Update Client.

5.2.3 Receiving update via FTP

In the configured polling interval, the FTP Update Client checks the FTP server to see if new files are available.

5.2.3.1 Downloads commands and resources for a unit from an FTP server

The update process checks the FTP server for a Command which is valid for it and then verifies if all the resources required for the update specified in the Command are available. The Commands and the resources for the unit(s) are downloaded from the FTP server. After a successful download, the Commands are deleted from the FTP server.

5.2.4 Finds valid Command file for a unit

Once downloaded, update application checks all the Commands relevant for the unit to find valid commands. Command validity is decided by the "BackgroundSystemGuid" and "UpdateIndex" parameters in the Command file. A command file is considered valid if for the same "BackgroundSystemGuid" as a last completed installation, the "UpdateIndex" is higher than the last completed installation. If a command file has a different "BackgroundSystemGuid" from a last completed installation, then the "UpdateIndex" is not considered and that command file is considered valid for the update. Only one command per unit is considered for the update, except if an older (not yet installed) command contains one or more pre- or post-installation actions; then that command is also executed in sequence. For the remaining commands only the feedback is sent (refer to chapter 5.2.8).

5.2.5 Creation of Update set

Once, the valid command file and required resources are available, an "Update set" is created based on the command file and the current file system structure on the unit. The Update Set is the set of differences between the file system structure in the command file and the current file system structure on the unit. This Update Set is used to perform the update of the system.

5.2.6 Order of update

1. If the Update Command contains pre-installation actions, they will be executed before anything else
2. If the Update Set created in the previous step contains an update of the Update application, then it is installed first.
3. The update of any other application(s) other than System Manager and their configuration is performed if it is part of the Update Set.
4. The last update is the update of System Manager and its configuration if it is part of the Update Set.
5. If the Update Command contains post-installation actions, they will be executed after everything else

5.2.7 Backup of current file system structure during update

The backup of the current file system structure is not a full backup. Based on the Update set, only the files and folders to be deleted and files to be updated (overwritten) are placed in backup. In case of failure of the update, using the Update Set and the backup, the system is restored to its state before the update was performed. This backup process is performed for each step in the section 5.2.6.
A detailed explanation of the process of backup and update is given below.

1. First step is to perform the backup of the current file system structure as explained above.
2. The next step is to copy all the new files to a temporary location on the unit.
3. Now all the new files are copied to the correct destination in the file system structure.
4. If this was successful, then the backup is finally deleted.

5.2.8 Feedback of update

Feedback for an update of a unit is provided at different instances during the update process. Initially, all the command files for a unit are analyzed and feedback is sent to the USB stick for each of the command files. If it is a slave unit and is registered to a master unit, the slave unit sends the feedback and the log files to the master unit which in turn will send the feedback to the USB stick.

- If a command file contains an UpdateIndex lower than the UpdateIndex of the last completed installation, then that command file is given a feedback of Update state: Ignored.

- If a command file contains an UpdateIndex equal to the UpdateIndex of the last completed installation, then that command file is given a feedback of Update state: Installed.
- If a command file contains an UpdateIndex greater than the UpdateIndex of the last completed installation, or contains a new "BackgroundSystemGuid", then that command file is given a feedback of Update state: Transferred.
- If an update was only partially installed, then the command file is given a feedback of Update state: PartiallyInstalled.

Once all the resources required for the update are available, then the feedback with Update state: Installing is sent. Once the complete update is installed and if the update was successful, then the feedback with Update state: Installed is sent. In case the update was unsuccessful, then a feedback with Update state: InstallationFailed is sent.
Upon completion of the update, all the feedback for the unit is sent to the repository (i.e. USB stick or FTP server) to the Feedback folder under a sub-folder for the unit. Along with the update feedback, all the archived log files for the unit are also transferred to the repository to the Feedback folder under a sub-folder for the unit.
Feedback is sent through all configured Update Clients, not just the one where the update was received. This means, if you configure an FTP and a USB Update Client, feedback will be uploaded to the FTP server when an update was received via FTP or USB; also whenever you plug in a USB stick, it will get all the feedback for all updates received via FTP or USB.

5.3 Rollback of an update

If an update fails or if the system is forced to shut down during an update (by turning off ignition), the update is rolled back partially or entirely. The rollback only contains the section of the update currently being executed (see chapter 5.2.6):

- If the update is stopped during the execution of pre-installation actions, all actions are executed to the end and then the update is aborted.
- If the update is stopped during the update of the Update application; this part is (in most cases) executed and then the update is aborted.
- If the update is stopped during the update of any other application (including System Manager), the update is rolled back, so none of the applications or their configuration have changed. Then the update is aborted.
- If the update is stopped during the execution of post-installation actions, all actions are executed to the end and then the update is aborted.

Rollbacks guarantee the "atomicity" of an update part: it is either installed or not installed. It will never happen that a few files of an application were updated but others were not. Like this the system will continue working even after a failed update.

5.4 Parked update

A parked update is an update which is set to be installed at the later time. When a TFT system receives an update with a time stamp set to a later time, it is stored on the TFT system if it is a valid parked update.

5.4.1 Validity of parked update

The validity of a parked update is checked in the same way as a normal update (refer to section 5.2.4).

5.4.1.1 Previously parked update when new parked update is received

If there is a previously parked update available on the unit, then, when a new parked update is received for the unit, the previously parked update is removed and a feedback that the previously parked update has been "Ignored" is sent. The new parked update is parked and the feedback that it has been "Transferred" is sent.

5.4.1.2 Previously parked update when a new normal update is received

If there is a previously parked update available on the unit, then, when a new normal update is received for the unit, the previously parked update is removed and a feedback that the previously parked update has been "Ignored" is sent.

5.4.2 Processing of parked update

A valid parked update is saved locally on the unit and a timer is started to know when it becomes valid. When the parked update becomes valid, it is processed like a normal update (refer to sections 5.2.5, 5.2.6, 5.2.8, 5.3)

5.5 Update feedback

The sections in this chapter explain the different update feedback states, their content and their meaning.

5.5.1 Created

The update has been created but was not yet sent to the unit. This is created by the software that generates the update (either USB Update Manager or icenter.Update).

5.5.2 Transferring

One or more intermediate update feedbacks might be sent for example when downloading an update via FTP. The "Transferring" feedback can be sent by an intermediate Unit (e.g. when a master is downloading an update over FTP and then forwarding it through Medi to a slave).

5.5.3 Transferred

The update has been successfully transferred to the destination Unit. This is usually reported by the Update Agent on the target Unit when the Update Command and all related resources are available.

5.5.4 Installing

The update is being installed on the unit. This state might be reported multiple times if the Update application is restarted during the installation (e.g. when updating Update.exe or its configuration or when System Manager has to be restarted – and thus restarts Update).

5.5.5 Installed

This is the final state that is reported when the update installed successfully. This feedback also contains the entire directory structure with all files, their hash and their state (i.e. "Up to date"). If an Update Command is sent to a Unit, but all files are already up to date, this state will still be reported, meaning "the given Update Command is already installed, I'm up to date."

5.5.6 Ignored

This state is sent if an Update Command contains an UpdateIndex lower than the UpdateIndex of the last completed installation. See also chapter 5.2.8.

5.5.7 Partially Installed

This state is sent if an update is aborted and the update (or a part of it) is rolled back. See also chapter 5.3. Partially installed updates are "final," they won't be installed again. This feedback also contains the entire directory structure with all files, their hash and their state.

5.5.8 Transfer Failed

This feedback is given if an intermediate Unit or the final Unit gets an Update Command but can't execute or forward it because resources are missing. This can happen either via FTP or USB if the "Resources" directory is completely missing or a needed "<hash>.rx" file is missing in the directory.

5.5.9 Installation Failed

This state is similar to "Partially Installed," but nothing was installed at all (e.g. when resources are missing on the Unit or the execution of pre-installation commands fails.