



033591
STOLPAN
STORE LOGISTICS AND PAYMENT WITH NFC

6th Framework IST project
Specific Targeted Research Project
IST-2005-2.5-8 – ICT for Networked Businesses

StoLPaN Mobile Track

D9

JAVA HOST FOR NFC APPLICATION

Due date of deliverable: Jul. 2007

Start date of project: 01.07.2006

Duration: 36 months

Organisation name of lead contractor for this deliverable:

Motorola (C0)

Final

Date:	04/11/2009
Author:	Zoltán Ábel
Version:	

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	PU
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission	
CO	Confidential, only for members of the consortium (including the Commission Services)	

The Consortium consist of

Participant No.	Participant Name	Short Name	Country
1	Motorola Ltd.	Motorola	HU
2	SafePay Systems Ltd.	SafePay	HU
3	Deloitte Ltd.	Deloitte	HU
4	Budapest Tech, John von Neumann Faculty of Informatics	BMF	HU
5	Auto-ID-Labs St. Gallen	Auto-ID-Lab	CH
6	BULL Ltd.	BULL	HU
7	Consult Hyperion	Chyp	UK
8	Fornax Plc.	Fornax	HU
9	NXP Austria Gmbh.	Philips A	AT
10	University of Technology and Economics	BME	HU
11	Banca Popolare di Vicenza	BPVI	IT
12	Libri Bookstores Ltd.	Libri	HU
13	Baker &McKenzie	BMcK	HU
14	Consorzio Triveneto S.P.A.	Constriv	IT
15	SUN Microsystems Ltd.	SUN	HU
16	T-Systems Hungary Ltd.	T-Systems	HU
17	NXP Italia Spa.	Philips IT	IT
18	Motorola Gmbh.	Motorola DE	DE
19	University of Rome	Cattid	IT
20	Ennova Research S.r.l	Ennova	IT
21	Sheffield Hallam University	Sheffield	UK
22	NXP Semiconductors France	NXP France	FR
23	Alkalmazás Fejlesztési Fórum Kft.,	AFF	HU

Table of Content

1. INTRODUCTION	5
1.1. PURPOSE.....	5
1.2. SCOPE.....	5
1.3. GOAL	5
1.4. AUDIENCE	5
1.5. TERMS AND DEFINITIONS	6
2. PACKAGE ORGANIZATION	8
3. PACKAGE HU.AFF.STOLPANHOST.HSC.....	10
3.1. CLASS COMPONENT	10
3.2. CLASS COMPONENTCONTEXT	12
3.3. CLASS TPSCWORKFLOWMANAGER.....	13
4. PACKAGE HU.AFF.STOLPANHOST.GUI.....	21
4.1. CLASS COMMANDS.....	21
4.2. CLASS SCREEN	24
5. PACKAGE HU.AFF.STOLPANHOST.HSC.INTERFACES	28
5.1. PACKAGE DESCRIPTION	28
5.2. INTERFACE ICOMPONENT.....	28
5.3. INTERFACE ITPSCWORKFLOWMANAGER	29
6. PACKAGE HU.AFF.STOLPANHOST.HSC.EVENTS.....	32
6.1. CLASS EVENTHANDLER	32
6.2. CLASS IEVENTREGISTRY.....	34
7. PACKAGE HU.AFF.STOLPANHOST.HSC.LOG	36
7.1. INTERFACE ILOGADAPTER	36
7.2. LOGENTRY	38
8. PACKAGE HU.AFF.STOLPANHOST.HSC.NFC	41
8.1. INTERFACE INFCMODESWITCHER	41
8.2. CLASS READERWRITERMODEDESCRIPTION	44
9. PACKAGE HU.AFF.STOLPANHOST.HSC.SCREENS.....	50
9.1. INTERFACE ISCREENMANAGER.....	50
10. PACKAGE HU.AFF.STOLPANHOST.HSC.SECUREELEMENT	53
10.1. INTERFACE ISECUREELEMENTADAPTER	53
11. PACKAGE HU.AFF.STOLPANHOST.HSC.STORAGE	55
11.1. INTERFACE ISTORAGEADAPTER.....	55
12. PACKAGE HU.AFF.STOLPANHOST.UTIL	60
12.1. INTERFACE PERSISTENT	60
12.2. CLASS BYTEUTIL	61
12.3. CLASS SLEEPER	63
12.4. CLASS RESOURCEBUNDLE	63

1. Introduction

1.1. Purpose

The purpose of this document is to describe the Java API interface of the StoLPaN Host Application.

This API is based on the Contactless Communication API (JSR-257), the Security And Trust Services API (JSR-177) and some other MIDP 2.0 basic API.

The public API is built on the StoLPaN Host Application which is a J2ME framework for third party vendors who will develop their applications to NFC and Smart MX based mobile phones. It provides some classes and interfaces for using the specific functions of these mobile phones. It hides the core J2ME APIs which let the third party vendors to focus their efforts to the application workflow.

This document gives the full technical specification of the API. It contains some restrictions about the classes and methods and sample code for how to use them.

1.2. Scope

The StoLPaN Host application outlined in this document builds upon the scope defined in the Requirements and General consideration chapter.

1.3. Goal

The HOST platform offers the possibility for the fast development of new functionalities, new designs and new experiences on a cost effective way by shortening the software development lifecycle and making application integration and migration easier to the NFC technology.

1.4. Audience

The intended audiences for this document are the project development teams, technical architects, testers and third party vendors.

1.5. Terms and Definitions

Term	Abbrev.	Type	Meaning
Application		Business	A Cardlet which will be stored in the Secure Element.
API		Technical	Application Programming Interface
Application Protocol Data Unit	APDU	Technical	Command sequence by which an application (Cardlet) can be controlled.
Connected Limited Device Configuration	CLDC	Technical	The CLDC defines the base set of application programming interfaces and a virtual machine for resource-constrained devices like mobile phones, pagers, and mainstream personal digital assistants.
Handset		Business	Mobile Phone
Java 2 Platform, Micro Edition	J2ME	Technical	The Java Platform, Micro Edition or Java ME (previously known as Java 2 Platform, Micro Edition or J2ME) is a specification of a subset of the Java platform aimed at providing a certified collection of Java APIs for the development of software for small, resource-constrained devices such as cell phones, PDAs and set-top boxes.
MIDlet		Technical	A MIDlet is a Java program for embedded devices, more specifically the Java ME virtual machine.
Mobile Information Device Profile	MIDP (API)	Technical	The MID Profile provides a standard run-time environment that allows new applications and services to be deployed dynamically on end-user devices.
NFC Data Exchange Format	NDEF	Technical	A message encapsulation format to exchange information e.g. an NFC Forum device and another NFC Forum device or an NFC Forum tag.
Near Field Communication	NFC	Technical	Near Field Communication (NFC) is a new, short-range wireless connectivity technology that evolved from a combination of existing contactless identification and interconnection technologies. Operating at 13.56 MHz and transferring data at up to 424 Kbits/second, NFC provides intuitive, simple, and safe communication between electronic devices. NFC is both a read and write technology. Communication between two NFC-compatible devices occurs when they are brought within four centimetres of one another: a simple wave or touch can establish an NFC connection which is then compatible with other known wireless technologies such as Bluetooth or Wi-Fi.. The underlying layers of NFC technology follow ISO, ECMA, and ETSI standards. Because the transmission range is so short, NFC-enabled transactions are inherently secure. Also, physical proximity of the device to the reader gives users the reassurance of being in control of the process.

OSGi		Technical	OSGi technology is the dynamic module system for Java. Reference: http://www.osgi.org/
Record Management System	RMS	Technical	A key subsystem of the Mobile Information Device Profile (MIDP) is the Record Management System (RMS), an application programming interface (API) that gives MIDP applications local, on-device data persistence. On most MIDP-enabled devices today, RMS is the only facility for local data storage -- few devices support a conventional file system.
RecordStore		Technical	A record store is an ordered collection of records. Records are not independent entities: each must belong to a record store, and all record access occurs through the record store. In fact, the record store guarantees that records are read and written atomically, with no possibility of data corruption. Record stores also maintain time-stamp and version information so applications can discover when a record store was last modified. The amount of memory available for record-based data storage varies from device to device. The MIDP specification requires devices to reserve at least 8K of non-volatile memory for persistent data storage.
SDK		Technical	Software Development Kit
Secure Element	SE	Business	Secure element (SE) is a smart card chip capable of storing multiple applications, e.g. SIM card, secure memory card or an additional embedded smart card chip in the NFC device. SE ID is the CPLC.
Service Provider	SP	Business	The Service Provider provides the business logic for the mobile handset NFC service.
Smart Card Application	Cardlet		A program written in Java language that runs on a JavaCard.
StoLPaN HOST		Business	The main MIDlet running on the user's Mobile phone. It contains a component platform to enable components based applications secure and easy NFC access.
Third Party Application	TPA	Business	Application provided by third party implementing the third party's business logic in Java ME Technology integrated to the StoLPaN Host.
Third Party Service Component	TPSC	Technical	TPSC is the technical implementation of the Third Party Application
User Interface	UI	Technical	Is a type of user interface which allows people to interact with a computer and computer-controlled devices. GUI: Instead of offering only text menus, or requiring typed commands: graphical icons, visual indicators or special graphical elements called "widgets", are presented.

2. Package Organization

Providing secure NFC based services was the center point at creation of this API. The *stolpanhost* package provides a mechanism for secure NFC communication and data handling. Along the *util* and *gui* sub packages the most important parts of the software packages are placed in the *hsc* (*host service component*) package.

Each package can be related to a well-defined component in the Host software environment. Therefore, most of the packages contain classes and interfaces, which are used to implement specific component functionality.

The following diagrams give an overview of the StoLPaN Host system. The first diagram shows the nesting relationship between the available packages.

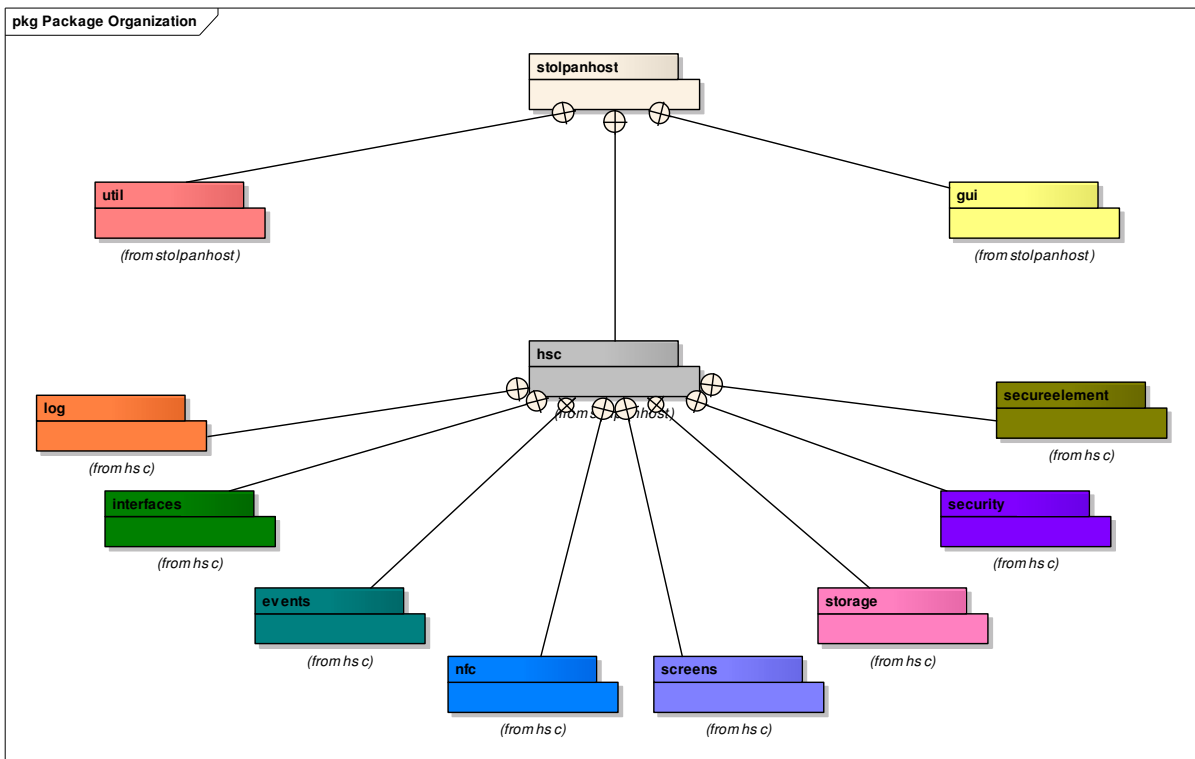


Figure 1. The Package Organization in the StoLPaN Host

The second diagram shows the associations between classes and interfaces. In the *hsc* package there are special classes belonging to the StoLPaN Host Platform. These classes create a component model which is the base of the platform. The components extend the base classes. They can be started, stopped and registered as services.

Every third party application has to implement one and only one workflow manager. This can be done by extending the *TpscWorkflowManager* abstract class. Therefore, they will also have all properties of StoLPaN Host Components. The *TpscWorkflowManager* extends the *Component* abstract class, which is the base of any component in the system. If third party applications may have other private components, these have to extend the *Component* class.

At application start up every third party applications is registered in a separate context which is the context of the TPSC. This context is accessible only for the workflow manager of the third party application who can register its other components dependencies into this context. A TPSC can use the components belonging to the platform via its specialized *TpscWorkflowManager* trough public interfaces.

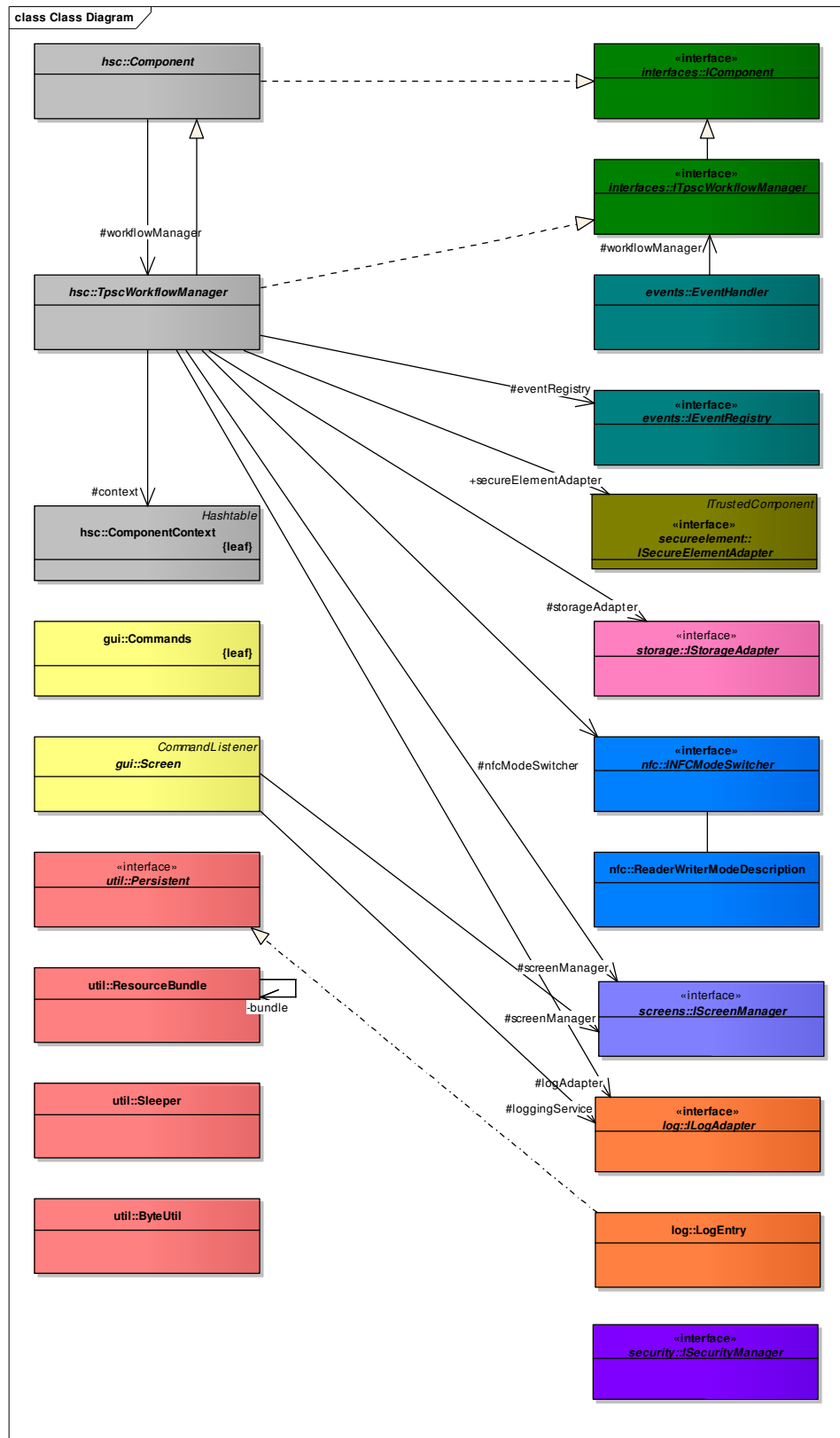


Figure 2 The Class and interface organization in the in the StoLPaN Host

3. Package hu.aff.stolpanhost.hsc

3.1. Class Component

3.1.1. Declaration

public abstract class Component implements IComponent

Method Summary	
public final boolean	isEnabled()
public final void	setEnabled(boolean)
public final boolean	isStarted()
public void	start(ComponentContext)
protected abstract void	startComponent(ComponentContext context)

3.1.2. Description

This is the abstract base class. All components must extend this class in order to obtain reference to ComponentContext. Components belongs to the Stolpan Host Framework or the third party software components (TPSC). Third party services can have one or more component. The workflow manager class is the first one.

Components (and other resources) have to be initialized at startup. This is done by using the startComponent method. Calling startComponent method every resource is initialized by the components abstract method specialized locally.

3.1.3. Methods

3.1.3.1. isEnabled()

Declaration:

public final boolean isEnabled()

Description:

Returns a Boolean variable which shows that the component is enabled or not. If a component is enabled it can be started and can registered to the component context.

Returns:

Boolean – if true the component can be registered and started.

3.1.3.2. setEnabled(boolean)

Declaration:

public void setEnabled(boolean enabled)

Description:

This setter method sets the Enabled attribute of a Component typed class. If the enabled attribute is set to true, the component can be started.

Parameters:

Enabled – the new value of the enabled attribute.

3.1.3.3. isStarted()

Declaration:

```
public boolean isStarted()
```

Description:

Getter method for started attribute.

Returns:

Boolean – if true, the component has already started.

3.1.3.4. start(ComponentContext)

Declaration:

```
public void start(ComponentContext context) throws Exception
```

Description:

Starts the component by calling the startComponent(ComponentContext) method and sets the component started attribute to true.

Parameters:

context – the componentContext object in which the component is registered in. The component object can access other components via this context.

Throws:

Exception – if any problem occurs at component startup. If this is thrown the component failed to start.

3.1.3.5. startComponent(ComponentContext)

Declaration:

```
protected abstract void startComponent(ComponentContext context) throws Exception
```

Description:

All classes which specialize the main Component class must implement this method. This method must handle all tasks related to component initialization. Components use this method to initialize themselves instead of the constructors. This is because of the component loader has to check if the component is enabled or not. If the component is not enabled that means it is no need to register, start and initialize it.

Parameters:

context – the componentContext object in which the component is registered in. The component object can access other components via this context.

Throws:

Exception - if any problem occurs at component startup. If this is thrown the component failed to start.

3.2. Class ComponentContext

3.2.1. Declaration

public final class ComponentContext extends java.util.Hashtable

Method Summary	
public IComponent	getThirdPartyComponent(java.lang.Class componentRef)
public void	registerThirdPartyComponent(String iface, Component service)

3.2.2. Description

This class implements a context for the whole application. Components can be registered in a context. At application start up the component loader registers all components of the framework to the context of the host. All third party workflow managers are registered in this context also.

When a component starts a context, it has to be set to it. If the component belongs to the framework then the host context will be set in. If the component is a third party workflow manager then a context of the third party service will be set in.

3.2.3. Methods

3.2.3.1. getThirdPartyComponent(java.lang.Class componentRef)

Declaration:

public IComponent getThirdPartyComponent(java.lang.Class componentRef)

Description:

Get a third party component reference from a context. Third party components can only access other third party components belongs to the same third party service. Third party components can be accessed via its component reference.

Parameters:

java.lang.Class componentRef – the class reference of the third party component to access.

Returns:

IComponent – third party component if the component is already registered to the context. The return value must be casted. If the component reference is not registered IllegalStateException is thrown.

Throws:

IllegalStateException – if there is not such component registered.

3.2.3.2. registerThirdPartyComponent(String iface, Component service)

Declaration:

public void registerThirdPartyComponent(String iface, Component service)

Description:

Registers a third party component to the context. A third party component can only register other components to its own component context.

Parameters:

String iface – the name of the interface of the component which will be registered. It is created by using the class.getName().

Component service – the component to register.

3.3. Class TpscWorkflowManager

3.3.1. Declaration

```
public abstract class TpscWorkflowManager extends Component
```

Field Summary	
protected IStorageAdapter	storageAdapter
protected ILogAdapter	logAdapter
protected IScreenManager	screenManager
protected INFCModeSwitcher	nfcModeSwitcher
protected IEventRegistry	eventRegistry
protected ISecureElementAdapter	secureElementAdapter
private Main	platform
protected ComponentContext	context

Method Summary	
public final void	exit()
protected final void	startComponent(ComponentContext context)
protected abstract void	startWorkflowComponent(ComponentContext context)
protected abstract void	stopWorkflowComponent()
public abstract void	activate()
public abstract void	inactivate()
public IStorageAdapter	getstorageAdapter()
public ILogAdapter	getlogAdapter()
public IScreenManager	getscreenManager()
public INFCModeSwitcher	getnfcModeSwitcher()
public IEventRegistry	geteventRegistry()
public ISecureElementAdapter	getsecureElementAdapter()
public void	setstorageAdapter(IStorageAdapter newVal)
public void	setlogAdapter(ILogAdapter newVal)
public void	setscreenManager(IScreenManager newVal)
public void	setnfcModeSwitcher(INFCModeSwitcher newVal)
public void	seteventRegistry(IEventRegistry newVal)
public void	setsecureElementAdapter(ISecureElementAdapter newVal)
public void	setsecurityManager(ISecurityManager newVal)

3.3.2. Description

Implements common workflow handling. Anything else workflow-related must be handled by its child classes. The host related components are set up to the workflow manager at start up.

3.3.3. Fields

3.3.3.1. storageAdapter

Declaration:

protected IStorageAdapter storageAdapter

Description:

Reference to the StorageAdapter Stolpan Host component.

3.3.3.2. logAdapter

Declaration:

protected ILogAdapter logAdapter.

Description:

Reference to the LogAdapter Stolpan Host component.

3.3.3.3. screenManager

Declaration:

protected IScreenManager screenManager

Description:

Reference to the ScreenManager Stolpan Host component.

3.3.3.4. nfcModeSwitcher

Declaration:

protected INFCModeSwitcher nfcModeSwitcher

Description:

Reference to the NFCModeSwitcher Stolpan Host component.

3.3.3.5. eventRegistry

Declaration:

protected IEventRegistry eventRegistry

Description:

Reference to the EventRegistryStolpan Host component.

3.3.3.6. secureElementAdapter

Declaration:

protected ISecureElementAdapter secureElementAdapter

Description:

Reference to the SecureElementAdapter Host component.

3.3.3.7. platform

Declaration:

private Main platform

Description:

Reference to the Stolpan Host midlet.

3.3.3.8. context

Declaration:

protected ComponentContext context

Description:

Reference to a Component Context of the given TPSC.

3.3.4. Methods

3.3.4.1. exit()

Declaration:

public final void exit()

Description:

This method is implemented in the abstract class TpscWorkflowManager. It calls the public void inactivate() method which developers of third party components can implement. Finally the exit method set the main menu of the Stolpan Host application

3.3.4.2. startComponent(ComponentContext context)

Declaration:

protected final void startComponent(ComponentContext context)

Description:

As TpscWorkflowManager is a component also, it has to implement this method. Workflow managers have special entry points called startWorkflowManager(ComponentContext context) which is called by this method. Third party developers in most of the cases are not needed to use or modify this method.

Parameters:

ComponentContext context – the context in which the workflow component of the third party service is registered in. Other components of the third party application can be registered in this context also.

3.3.4.3. activate()

Declaration:

public abstract void activate()

Description:

This method is the entry point of all TPSCs. If the user selects the third party component then the framework calls this method of the workflow manager. Important: developers doesn't have to implement here the constructor of the workflow or the whole third party component. These have to be implemented at the protected void startWorkflowComponent(ComponentContext arg0) method of the workflow manager (inherited from TpscWorkflowManager).

3.3.4.4. **inactivate()**

Declaration:

```
public abstract void inactivate()
```

Description:

Every TPSC has to have an exit point. This method is the exit point of all TPSCs. The inactivation means that the application gets back to the main menu (implemented by default). Any other task related to application inactivation must be implemented here.

3.3.4.5. **getstorageAdapter()**

Declaration:

```
public IStorageAdapter getstorageAdapter()
```

Description:

Getter for storageAdapter host component.

Returns:

The singleton storageAdapter component of the host application.

3.3.4.6. **getlogAdapter()**

Declaration:

```
public ILogAdapter getlogAdapter()
```

Description:

Getter for logAdapter host component.

Returns:

The singleton logAdapter component of the host application.

3.3.4.7. **getscreenManager()**

Declaration:

```
public IScreenManager getscreenManager()
```

Description:

Getter for screenManagerAdapter host component.

Returns:

The singleton screenManagerAdapter component of the host application.

3.3.4.8. **getnfcModeSwitcher()**

Declaration:

```
public INFModeSwitcher getnfcModeSwitcher()
```

Description:

Getter for nfcModeSwitcher host component.

Returns:

The singleton nfcModeSwitcher component of the host application.

3.3.4.9. geteventRegistry()**Declaration:**

```
public IEventRegistry geteventRegistry()
```

Description:

Getter for eventRegistry host component.

Returns:

The singleton eventRegistry component of the host application.

3.3.4.10. getsecureElementAdapter()**Declaration:**

```
public ISecureElementAdapter getsecureElementAdapter()
```

Description:

Getter for secureElementAdapter host component.

Returns:

The singleton secureElementAdapter component of the host application.

3.3.4.11. getsecurityManager()**Declaration:**

```
public ISecurityManager getsecurityManager()
```

Description:

Getter for securityManager host component.

Returns:

The singleton securityManager component of the host application.

3.3.4.12. setstorageAdapter(IStorageAdapter newVal)**Declaration:**

```
public void setstorageAdapter(IStorageAdapter newVal)
```

Description:

Setter for storageAdapter host component used by component loader.

Parameters:

IStorageAdapter newVal - singleton storageAdapter host component.

3.3.4.13. setlogAdapter(ILogAdapter newVal)**Declaration:**

```
public void setlogAdapter(ILogAdapter newVal)
```

Description:

Setter for logAdapter host component used by component loader.

Parameters:

ILogAdapter newVal - singleton logAdapter host component.

3.3.4.14. setscreenManager(IScreenManager newVal)**Declaration:**

```
public void setscreenManager(IScreenManager newVal)
```

Description:

Setter for screenManager host component used by component loader.

Parameters:

IScreenManager newVal - singleton screenManager host component.

3.3.4.15. setnfcModeSwitcher(INFCModeSwitcher newVal)**Declaration:**

```
public void setnfcModeSwitcher(INFCModeSwitcher newVal)
```

Description:

Setter for nfcModeSwitcher host component used by component loader.

Parameters:

INFCModeSwitcher newVal - singleton nfcModeSwitcher host component.

3.3.4.16. seteventRegistry(IEventRegistry newVal)**Declaration:**

```
public void seteventRegistry(IEventRegistry newVal)
```

Description:

Setter for eventRegistry host component used by component loader.

Parameters:

IEventRegistry newVal - singleton eventRegistry host component.

3.3.4.17. setsecureElementAdapter(ISecureElementAdapter newVal)**Declaration:**

```
public void setsecureElementAdapter(ISecureElementAdapter newVal)
```

Description:

Setter for secureElementAdapter host component used by component loader.

Parameters:

ISecureElementAdapter newVal - singleton secureElementAdapter host component.

3.3.4.18. setsecurityManager(ISecurityManager newVal)**Declaration:**

```
public void setsecurityManager(ISecurityManager newVal)
```

Description:

Setter for securityManager host component used by component loader.

Parameters:

ISecurityManager newVal - singleton securityManager host component.

4. Package hu.aff.stolpanhost.gui

4.1. Class Commands

4.1.1. Declaration

```
public final class Commands
```

Field Summary	
public static final Command	OK
public static final Command	BACK
public static final Command	CANCEL
public static final Command	EXIT
public static final Command	LOGIN
public static final Command	SELECT
public static final Command	HELP
public static final Command	CLEAR
public static final Command	CHANGE
public static final Command	SAVE

4.1.2. Description

Contains static commands which can be used by TPSC developers to manage their own GUI. They do not have to create their own commands for the GUI events already specified in this class.

4.1.3. Fields

4.1.3.1. OK

Declaration:

public static final Command OK

Description:

Static OK command.

4.1.3.2. BACK

Declaration:

public static final Command BACK

Description:

Static BACK command.

4.1.3.3. CANCEL

Declaration:

public static final Command CANCEL

Description:

Static CANCEL command.

4.1.3.4. EXIT

Declaration:

public static final Command EXIT

Description:

Static EXIT command.

4.1.3.5. LOGIN

Declaration:

public static final Command LOGIN

Description:

Static LOGIN command.

4.1.3.6. SELECT

Declaration:

public static final Command SELECT

Description:

Static SELECT command.

4.1.3.7. HELP

Declaration:

public static final Command HELP

Description:

Static HELP command.

4.1.3.8. CLEAR

Declaration:

public static final Command CLEAR

Description:

Static CLEAR command.

4.1.3.9. CHANGE

Declaration:

public static final Command CHANGE

Description:

Static CHANGE command.

4.1.3.10. SAVE

Declaration:

public static final Command SAVE

Description:

Static SAVE command.

4.2. Class Screen

4.2.1. Declaration

public abstract class Screen implements CommandListener

Field Summary	
protected IScreenManager	screenManager
protected ILogAdapter	loggingService
protected Displayable	displayable
protected boolean	history
protected boolean	editingInProgress

Method Summary	
public final Displayable	getDisplayable()
public boolean	keepHistory()
public boolean	isEditingInProgress()
public final void	show()
public final void	commandAction(Command c, Displayable d)
protected abstract void	showScreen()
protected abstract void	handleCommand(Command c)

4.2.2. Description

Abstract screen class. All screens (forms, canvases, lists, etc.) must extend this class.

IMPORTANT: Child screens must initialize (assemble) themselves in their constructors, but they must not call references to other components, since these references will be set by the setScreenManager() call.

4.2.3. Fields

4.2.3.1. screenManager

Declaration:

protected IScreenManager screenManager

Description:

Screen manager of the Stolpan Host application. It is set by the platform at startup.

4.2.3.2. loggingService

Declaration:

protected ILogAdapter loggingService

Description:

Logging adapter component of the Stolpan Host application. It is set by the platform at startup.

4.2.3.3. displayable

Declaration:

protected Displayable displayable

Description:

Displayable variable to change the actual screen.

4.2.3.4. history

Declaration:

protected boolean history

Description:

Flag to show that the screen belongs to the history tree.

4.2.3.5. editingInProgress

Declaration:

protected boolean editingInProgress

Description:

Not used already. May be use for a flag to show that editing is in progress so please not to show it.

4.2.4. Methods

4.2.4.1. getDisplayable()

Declaration:

public final Displayable getDisplayable()

Description:

Returns the actual displayable object.

Returns:

Displayable – the actual displayable object.

4.2.4.2. keepHistory()**Declaration:**

```
public boolean keepHistory()
```

Description:

Getter for history.

Returns:

Boolean - Flag to show that the screen belongs to the history tree.

4.2.4.3. isEditingInProgress()**Declaration:**

```
public boolean isEditingInProgress()
```

Description:

Getter for editingInProgress.

Returns:

Not implemented yet. Shows that editing is in progress so one can use this method to delay showing a screen.

4.2.4.4. show()**Declaration:**

```
public final void show()
```

Description:

Shows the actual screen on the display.

4.2.4.5. commandAction(Command c, Displayable d)**Declaration:**

```
public final void commandAction(Command c, Displayable d)
```

Description:

Handles the command. Mandatory method of the CommandListener interface.

Parameters:

Command c – the command to handle.

Displayable d – the displayable object containing the command.

4.2.4.6. showScreen()**Declaration:**

```
protected abstract void showScreen()
```

Description:

Every screens has to implement this method. It is run when the show method is called. Screens can implement their own tasks related to showing the specific screens here.

4.2.4.7. handleCommand(Command c)

Declaration:

protected abstract void handleCommand(Command c)

Description:

Abstract method. Every screen has to implement it. It does the work of the commandAction(Command c, Displayable d) method. It has to handle every user interaction.

Parameters:

Command c – the command to handle.

5. Package hu.aff.stolpanhost.hsc.interfaces

5.1. Package description

This package contains only interfaces. Applications can use interfaces of this package to use host specific services. These are some basic interfaces used by third party applications.

5.2. Interface IComponent

5.2.1. Declaration

```
public interface IComponent
```

Method Summary	
public boolean	isEnabled
public boolean	isStarted()
public void	start(ComponentContext context)

5.2.2. Description

Interface of the abstract Component class. All components must extend this class in order to obtain reference to ComponentContext. The interface is used by Stolpan Host Framework developers and doesn't need to be used by third party vendors.

5.2.3. Methods

5.2.3.1. isEnabled()

5.2.3.2. Declaration:

```
public boolean isEnabled()
```

Description:

This getter method retrieves the Enabled attribute of a Component typed class which indicates that the component can be started or not.

Returns:

Boolean value of the Enabled variable.

5.2.3.3. Declaration:

```
public void setEnabled(boolean enabled)
```

Description:

This setter method sets the Enabled attribute of a Component typed class. If the enabled attribute is set to true, the component can be started.

Parameters:

Enabled – the new value of the enabled attribute.

5.2.3.4. isStarted()**5.2.3.5. Declaration:**

```
public boolean isStarted()
```

Description:

Getter method for started attribute.

Returns:

Boolean – if true, the component has already started.

5.2.3.6. start(ComponentContext)**Declaration:**

```
public void start(ComponentContext context) throws Exception
```

Description:

Starts the component by calling the startComponent(ComponentContext) method and sets the component started attribute to true.

Parameters:

context – the componentContext object in which the component is registered in. The component object can access other components via this context.

Throws:

Exception if any problem occurs at component startup. If this is thrown the component failed to start.

5.3. Interface ITpscWorkflowManager**5.3.1. Declaration**

```
public interface ITpscWorkflowManager extends IComponent
```

Method Summary	
public abstract void	activate()
public abstract void	inactivate()
public void	exit()

5.3.2. Description

Interface for the parent class for the workflow manager class of a TPSC. A Workflow manager class is the main class of the TPSC. Every TPSC has to extend the `TpscWorkflowManager` abstract class which implements this interface. If the TPSC has its own interface that interface has to extend the `ITpscWorkflowManager` interface.

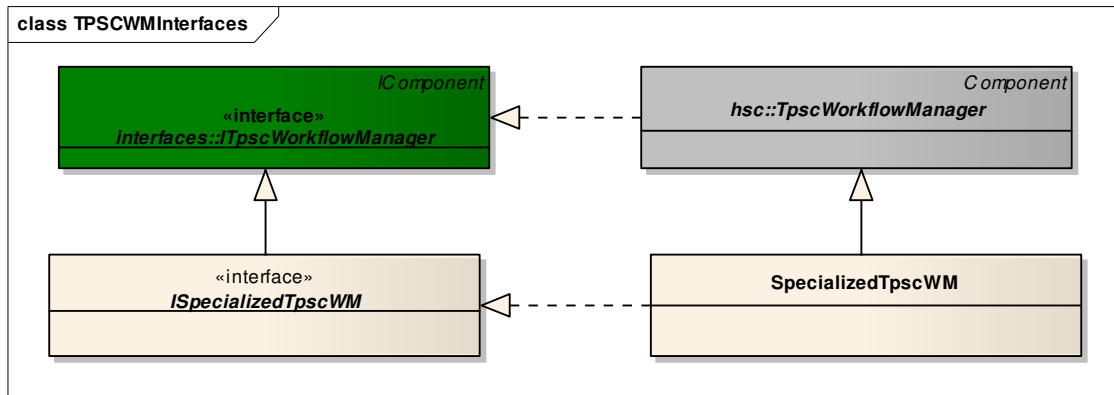


Figure 3. Tpsc Class Relations

5.3.3. Methods

5.3.3.1. activate()

Declaration:

```
public abstract void activate()
```

Description:

Every TPSC has to have an entry point. This method is the entry point of all TPSCs. If the user selects the special third party application from the application list then the framework calls this method of the workflow manager. Important: developers do not have to implement here the constructor of the workflow and the whole third party component. This has to be implemented at the protected void `startWorkflowComponent(ComponentContext arg0)` method of the workflow manager (inherited from `TpscWorkflowManager`).

5.3.3.2. exit()

Declaration:

```
public void exit();
```

Description:

This method is implemented in the abstract class `TpscWorkflowManager`. It calls the public void `inactivate()` method which developers of third party components can implement adding additional exit tasks. Finally the main screen will be loaded (implemented by default)

5.3.3.3. inactivate()

Declaration:

```
public abstract void inactivate()
```

Description:

Every TPSC has to have an exit point. This method is the exit point of all TPSCs. The inactivation means that the application gets back to the main menu (implemented by default). Any other inactivation task must be implemented here.

6. Package hu.aff.stolpanhost.hsc.events

6.1. Class EventHandler

6.1.1. Declaration

```
public abstract class EventHandler
```

Method Summary	
public final void	setWorkflowManager(ITpscWorkflowManager workflowManager)
public final void	setWorkflowManager(ITtrustedWorkflowManager workflowManager)
public final void	handle(String eventName, Persistent data)
protected abstract void	handleEvent(String eventName, Persistent data)

6.1.2. Description

Abstract event handler. All event handlers must extend this class. Event handlers provide a simple way of handling asynchronous events generated by low-level service components. Currently there are event handlers implemented for NFC communication.

Event handling in card emulation mode: card emulation mode has one special event. If a transaction happens, the TransactionListener (JSR-257) triggers the Host application about it. The framework detects this triggering and generates the appropriate event. It searches for event handlers who registered to that event before. If there is such an event handler the framework calls its handle method so the third party component can manage the event itself.

6.1.3. Methods

6.1.3.1. setWorkflowManager(ITpscWorkflowManager workflowManager)

Declaration:

```
public final void setWorkflowManager(ITpscWorkflowManager workflowManager)
```

Description:

Sets the reference of the third party workflow manager to the event handler. The event handler and the workflow manager must belong to the same third party service. Mostly but not exclusively the event handler needs the workflow manager to handle the event.

Parameters:

ITpscWorkflowManager workflowManager – the workflow manager belonging to the third party service.

6.1.3.2. setWorkflowManager(ITrustedWorkflowManager workflowManager)

Declaration:

```
public final void setWorkflowManager(ITrustedWorkflowManager workflowManager)
```

Description:

Sets the reference of the trusted workflow manager to the event handler. The event handler and the workflow manager in this case belong to the host framework. Mostly but not exclusively the event handler needs the workflow manager to handle the event.

Parameters:

ITrustedWorkflowManager workflowManager - the workflow manager belonging to the host framework.

6.1.3.3. handle(String eventName, Persistent data)

Declaration:

```
public final void handle(String eventName, Persistent data) throws Exception
```

Description:

Handles the event identified by the eventName. It calls the handleEvent(String eventName, Persistent data) method. Checks that an event handler is registered or not.

Parameters:

String eventName – event identifier. It can be special Stolpan Host Framework event.

Occupied event names are the following:

- EXTERNALREADERDETECTEDEVENT - hu.aff.stolpanhost.hsc.nfc, this event is occurs if the TransactionListener (JSR-257) is active.

Persistent data – data which belongs to the event. I.e if the event is a tag read the data parameter contains the data on the tag.

Throws:

IllegalStateException – if event handler not already registered.

6.1.3.4. handleEvent(String eventName, Persistent data)

Declaration:

```
protected abstract void handleEvent(String eventName, Persistent data) throws Exception
```

Description:

Handles the event identified by the eventName. Third party developers can implement the tasks what they want to do do if the event occurs.

Parameters:

String eventName – event identifier. It can be a special Stolpan Host Framework event.

Persistent data – data which belongs to the event. I.e if the event is a tag read the data parameter contains the data on the tag.

Throws:

IllegalStateException – if event handler was not already registered.

6.2. Class IEventRegistry

6.2.1. Declaration

```
public interface IEventRegistry
```

Method Summary	
public void	handleEvent(String eventName, Persistent eventData)
public void	addEventHandler(ITpscWorkflowManager iTpscWorkflowManager, String eventName, EventHandler eventHandler)
public void	removeEventHandler(ITpscWorkflowManager iTpscWorkflowManager, String eventName)

6.2.2. Description

IEventRegistry provides interface for EventRegistry singleton Host component. This component is the repository of the event handlers created by third party services and Stolpan host framework. If an event handler is created it must be registered to the event registry via this interface. The event is identified by the event name. When an event occurs the handlers of the low-level processes are searching for event handlers using EvenRegistry host component. This component looks for the event handler which is registered to that event name.

6.2.3. Methods

6.2.3.1. addEventHandler(ITpscWorkflowManager iTpscWorkflowManager, String eventName, EventHandler eventHandler)

Declaration:

```
public void addEventHandler(ITpscWorkflowManager iTpscWorkflowManager, String eventName,
    EventHandler eventHandler)
```

Description:

Registers an event handler created by a third party service. The event handler will be called by the framework if the event identified by the eventName parameter occurs.

Parameters:

ITpscWorkflowManager iTpscWorkflowManager – third party workflow manager. It identifies the third party service.

String eventName – assigns the event handler to an event. Event handler is identified by the name of the event.

EventHandler eventHandler – Reference of the event handler to register. If the event identified by the eventName parameter occurs, the framework calls the handle method of the eventHandler.

6.2.3.2. removeEventHandler(ITpscWorkflowManager iTpscWorkflowManager, String eventName)

Declaration:

```
public void removeEventHandler(ITpscWorkflowManager iTpscWorkflowManager, String eventName)
```

Description:

Removes the event handler identified by the event name and created by a third party service. The event handler can not be accessed by the framework after removing it. If third party service needs again the event handler it must be added again.

Parameters:

ITpscWorkflowManager iTpscWorkflowManager - third party workflow manager. It identifies the third party service.

String eventName - assigns the event handler to an event. The event handler is identified by the name of the event.

7. Package hu.aff.stolpanhost.hsc.log

7.1. Interface ILogAdapter

7.1.1. Declaration

```
public interface ILogAdapter
```

Method Summary	
public void	writeToLog(String logMessage, String logLevel, String logFileName)
public void	readFromLog(String logFileName)
public void	deleteLog(String logFileName)
public long	getLogSize(String logFileName)

7.1.2. Description

Interface for Logger component that handles notifications, debugs, warnings and errors. The storage for the log can be the RMS, file system or a removable data storage. Currently only logging to RMS service is implemented.

Third party services may use this logging functions. Every third party service can have one or more log message list. TPSCs can write, read, delete their log with the help of the methods defined in this interface.

7.1.3. Methods

7.1.3.1. writeToLog(String logMessage, String LogLevel, String LogFileName)

Declaration:

```
public void writeToLog(String logMessage, String logLevel, String logFileName) throws
LogSaveNotPossibleException, LogFileFullException
```

Description:

This method logs a message associated with a specific service and log level to the given log file. The log is stored in RMS.

Parameters:

String logMessage – message to store in the log file.

String logLevel – level of the log. It can be DEBUG, INFO, or ERROR. The levels are static variables found in class LogEntry.

String logFileName – name of the file where the message is logged. A third party service later can read this log file.

Throws:

LogSaveNotPossibleException – if there is any problem about the saving the log message.

LogFileFullException – if the log file is full.

7.1.3.2. readFromLog(String logFileName)

Declaration:

```
public LogEntry [] readFromLog(String logFileName) throws LogFileNotFoundException,
DataNotReadableException
```

Description:

Reads the full content of a log file. It returns a logEntry array containing all previously logged records read from the given log file.

Parameters:

String logFileName - name of the file where the message is read.

Returns:

LogEntry [] – array which contains all entry of the log file.

Throws:

LogFileNotFoundException – if the logFileName parameter is invalid.
DataNotReadableException – if any error occurs during the read process.

7.1.3.3. deleteLog(String logFileName)

Declaration:

```
public void deleteLog(String logFileName) throws LogFileNotFoundException
```

Description:

Deletes the given log file.

Parameters:

String logFileName - name of the file to delete.

Throws:

LogFileNotFoundException – if the logFileName parameter is invalid.

7.1.3.4. getLogSize(String logFileName)

Declaration:

```
public long getLogSize(String logFileName) throws LogFileNotFoundException
```

Description:

Returns the size of the log file identified by the logFileName parameter.

Parameters:

String logFileName – identifies the given log file.

Returns:

long – the size of the log file in bytes.

Throws:

LogFileNotFoundException – if the logFileName parameter is invalid.

7.2. LogEntry

7.2.1. Declaration

public class LogEntry implements Persistent

Field Summary	
public static final String	INFO
public static final String	DEBUG
public static final String	WARNING
public static final String	ERROR

Method Summary	
public String	getMessage()
public void	setMessage(String message)
public String	getDate()
public void	setDate(String date)
public String	getLevel()
public void	setLevel(String level)

7.2.2. Description

Represents a log message. Reading or writing a log is possible using LogEntry objects.

7.2.3. Fields

7.2.3.1. INFO

Declaration:

```
public static final String INFO = "INFO";
```

Description:

INFO log level

7.2.3.2. DEBUG

Declaration:

```
public static final String DEBUG = "DEBUG";
```

Description:

DEBUG log level

7.2.3.3. WARNING

Declaration:

```
public static final String WARNING = "WARNING";
```

Description:

WARNING log level

7.2.3.4. ERROR

Declaration:

```
public static final String ERROR = "ERROR";
```

Description:

ERROR log level

7.2.4. Methods

7.2.4.1. getMessage()

Declaration:

```
public String getMessage()
```

Description:

Returns the message which is the payload of the LogEntry object.

Returns:

String – the message part of the log

7.2.4.2. setMessage(String message)

Declaration:

```
public void setMessage(String message)
```

Description:

Sets the message which is the payload of the LogEntry object.

Parameters:

String message – the message part of the log

7.2.4.3. getDate()

Declaration:

```
public String getDate()
```

Description:

Returns the creation date of the log message.

Returns:

String - The date part of the log as String. The format of the date is the standard J2ME date format. The format is: dow mon dd hh:mm:ss zzz yyyy where:

- dow is the day of the week (Sun, Mon, Tue, Wed, Thu, Fri, Sat).

- `mon` is the month (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec).
- `dd` is the day of the month (01 through 31), as two decimal digits.
- `hh` is the hour of the day (00 through 23), as two decimal digits.
- `mm` is the minute within the hour (00 through 59), as two decimal digits.
- `ss` is the second within the minute (00 through 61, as two decimal digits.
- `zzz` is the time zone (and may reflect daylight savings time). If time zone information is not available, then `zzz` is empty - that is, it consists of no characters at all.
- `yyyy` is the year, as four decimal digits.

7.2.4.4. setDate(String date)

Declaration:

```
public void setDate(String date)
```

Description:

Sets the creation date of the log message.

Parameters:

String date – the date of the log message as String. The date format is the standard J2ME String date format.

7.2.4.5. getLevel()

Declaration:

```
public String getLevel()
```

Description:

Returns the level of the log message.

Returns:

String – one of the log levels declared in LogEntry class.

7.2.4.6. setLevel(String level)

Declaration:

```
public void setLevel(String level)
```

Description:

Sets the level of the log message.

Parameters:

String level – the level of the log message. It can be one of the levels declared in the LogEntry class.

8. Package hu.aff.stolpanhost.hsc.nfc

8.1. Interface INFCModeSwitcher

8.1.1. Declaration

```
public interface INFCModeSwitcher
```

Field Summary	
public static final String	IDLE_MODE_LOCK
public static final String	PROXY_MODE_LOCK
public static final String	READER_WRITER_MODE_LOCK
public static final String	CARD_EMULATION_MODE_LOCK
public static final String	EXTERNALREADERDETECTEDEVENT
public static String	APDU_COMMUNICATION_LOCK

Method Summary	
public void	setReaderWriterMode(ReaderWriterModeDescription readerWriterModeDescription)
public void	setCardEmulationMode()
public void	setProxyMode()
public void	setidleMode()

8.1.2. Description

Interface for the NFCModeSwitcher Stolpan host framework component. The component can manage the card emulation mode, reader/writer mode and a virtual mode what is called proximity mode or idle mode.

In Card emulation mode the internal secure element is visible for outer world. Terminals can send APDU command to it and it sends response. Communication is done via NFC chipset.

In Reader/Writer mode the application uses the NFC chipset to read or write NFC-Forum compatible tags or smart posters.

In Proximity mode the application can send APDU commands to the internal secure element and can receive responses.

Idle mode means that no other modes are set in. The application do not listen to any TransactionListener or TargetListener (defined by JSR-257) and can not send APDU commands to the internal secure element. At Stolpan Host Application start up the idle mode is the default one until a third party component does not need to use another mode.

In Stolpan Host framework only one mode can be set simultaneously. If another mode is set by the third party component then the Host exits the actual mode and sets the new mode.

8.1.3. Fields

8.1.3.1. IDLE_MODE_LOCK

Declaration:

```
public static final String IDLE_MODE_LOCK
```

Description:

Lock object for idle mode. Used as mode name and thread synchronization lock.

8.1.3.2. PROXY_MODE_LOCK

Declaration:

```
public static final String PROXY_MODE_LOCK
```

Description:

Lock object for proxy mode. Used as mode name and thread synchronization lock.

8.1.3.3. READER_WRITER_MODE_LOCK

Declaration:

```
public static final String READER_WRITER_MODE_LOCK
```

Description:

Lock object for reader/writer mode. Used as mode name and thread synchronization lock.

8.1.3.4. CARD_EMULATION_MODE_LOCK

Declaration:

```
public static final String CARD_EMULATION_MODE_LOCK
```

Description:

Lock object for card emulation mode. Used as mode name and thread synchronization lock.

8.1.3.5. EXTERNALREADERDETECTEDEVENT

Declaration:

```
public static final String EXTERNALREADERDETECTEDEVENT
```

Description:

Lock object for External reader detect event.

8.1.3.6. MANUAL_SWITCHING_MODE

Declaration:

```
public static final int MANUAL_SWITCHING_MODE
```

Description:

Constant for manual (hard coded) mode switching

8.1.3.7. APDU_COMMUNICATION_LOCK

Declaration:

```
public static String APDU_COMMUNICATION_LOCK
```

Description:

Lock object for thread synchronization with {@link SecureElementAdapter}

8.1.4. Methods

8.1.4.1. setReaderWriterMode(ReaderWriterModeDescription readerWriterModeDescription)

Declaration:

```
public void setReaderWriterMode(ReaderWriterModeDescription readerWriterModeDescription) throws MissingParameterException
```

Description:

Sets the Stolpan Host application to Reader/Writer mode. The application will be listening for NFC-Forum compatibles tags. Third party services must set the descriptor parameter because if an NFC-Forum compatible tag is detected the framework has to execute the right process to handle the read or write. The descriptor defines what to do if a tag is detected. I.e. read or write the tag, what kind of connection to use, which sector to read etc.

On Motorola L7 phones only Mifare Standard tags are available for reading or writing.

On Nokia 6131 Mifare UltraLight tags are also supported. If the descriptor is not set correctly MissingParameterException is thrown by the framework.

Initiating this mode automatically ends any previous NFC modes set in the application.

Parameters:

ReaderWriterModeDescription readerWriterModeDescription – contains parameters for the framework. It signals to the framework what to do if a target is detected.

Throws:

MissingParameterException – if the descriptor not correctly set by the third party service.

8.1.4.2. setCardEmulationMode()

Declaration:

```
public void setCardEmulationMode()
```

Description:

Sets the application and NFC hardware to Card emulation mode. In this mode the framework sets the secure element visible for the external terminals. This way a terminal can communicate with the selected cardlet using APDU commands or read or write to the internal Mifare sectors. Initiating this mode automatically ends the previous NFC mode set in the application.

8.1.4.3. setProxyMode()

Declaration:

```
public void setProxyMode()
```

Description:

Sets the application and NFC hardware to Proximity mode. This mode is not specified by the NFC forum as a separate mode but the Stolpan Host application handles it as an individual mode.

In this mode the third party services can communicate with their cardlets using APDU commands. They can send and receive APDUs. Note that there is no way to run other modes if there is already any communication in progress between the secure element and the Stolpan Host application. Initiating this mode automatically ends the previous NFC mode set in the application.

8.1.4.4. setIdleMode()

Declaration:

```
public void setIdleMode()
```

Description:

Sets the application and NFC hardware to Idle mode. This mode is not specified by the NFC forum as a separate mode but the Stolpan Host application handles it as an individual mode. In this mode there is no registered TargetListener and TransactionListener (defined in JSR-257) and no communication available with the secure element. This is a passive mode which means that no other modes are active in the Stolpan Host application.

8.2. Class ReaderWriterModeDescription

8.2.1. Declaration

```
public class ReaderWriterModeDescription
```

Field Summary	
public static final int	MIFARE_STANDARD_CONNECTION
public static final int	MIFARE_ULTRALIGHT_CONNECTION
public static final int	NDEF_CONNECTION

Method Summary	
public boolean	isreaderMode()
public void	setreaderMode(boolean newMode)
public byte[]	gettagKey()
public void	settagKey(byte[] newTagKey)
public int	gettagStartBlock()
public void	settagStartBlock(int startBlock)
public byte[]	gettagDataToWrite()
public void	settagDataToWrite(byte[] tagDataToWrite)
public int	getOffset()
public int	gettagConnectionType()
public void	setOffset(int offSet)
public void	settagConnectionType(int tagConnectionType)

8.2.2. Description

Repository class for NFC modes related settings. It contains parameters for the Reader / Writer mode tasks. TPSC developers have to use this parameter set to configure the Reader / Writer modul of the Stolpan application.

8.2.3. Fields

8.2.3.1. MIFARE_STANDARD_CONNECTION

Declaration:

```
public static final int MIFARE_STANDARD_CONNECTION
```

Description:

MIFARE_STANDARD_CONNECTION tag connection mode

8.2.3.2. MIFARE_ULTRALIGHT_CONNECTION

Declaration:

```
public static final int MIFARE_ULTRALIGHT_CONNECTION
```

Description:

MIFARE_ULTRALIGHT_CONNECTION tag connection mode

8.2.3.3. NDEF_CONNECTION

Declaration:

```
public static final int NDEF_CONNECTION
```

Description:

NDEF_CONNECTION tag connection mode

8.2.4. Methods

8.2.4.1. isreaderMode()

Declaration:

```
public boolean isreaderMode()
```

Description:

returns that the actual mode is reader or writer

Returns:

Boolean – true if the application is set to read

8.2.4.2. setreaderMode(boolean newMode)

Declaration:

```
public void setreaderMode(boolean newMode)
```

Description:

Sets the application to read or write if a tag is detected. Sets the actual direction of the operation.

Parameters:

boolean newMode – the new value to set. True if the next task is to read if a tag is detected. False if the next task is write.

8.2.4.3. gettagKey()

Declaration:

```
public byte[] gettagKey()
```

Description:

Getter for the actual tag key.

Returns:

byte[] – the actual set key. The Stolpan Host application will use this key to read or write the given sectors.

8.2.4.4. `settagKey(byte[] newTagKey)`

Declaration:

```
public void settagKey(byte[] newTagKey)
```

Description:

Sets a key for the Stolpan Host application. It will use this key to read or write the given sectors on a tag.

Parameters:

`byte[] newTagKey` – the sector key in byte array format

8.2.4.5. `gettagStartBlock()`

Declaration:

```
public int gettagStartBlock()
```

Description:

Returns the start block previously set. Reading or writing a tag will be start from this block.

Returns:

`int` – the number of the block

8.2.4.6. `settagStartBlock(int startBlock)`

Declaration:

```
public void settagStartBlock(int startBlock)
```

Description:

Sets the position of the block where the reading or writing must start.

Parameters:

`int` – block number.

8.2.4.7. `gettagDataToWrite()`

Declaration:

```
public byte[] gettagDataToWrite()
```

Description:

Returns the data that is set to write by the Stolpan Host application if a tag is detected.

Returns:

`byte[]` – data to write in byte array format.

8.2.4.8. `settagDataToWrite(byte[] tagDataToWrite)`

Declaration:

```
public void settagDataToWrite(byte[] tagDataToWrite)
```

Description:

Sets the data to be written by the Stolpan Host application if a tag is detected.

Parameters:

`byte[] tagDataToWrite` -data to write to the tag.

8.2.4.9. `getOffset()`

Declaration:

```
public int getOffset()
```

Description:

Returns the previously set offset value . Data will be written from start block to the offset number of blocks if the write mode is set. If read mode is set data will be read from start block to the offset block.

Returns:

int – value of the offset.

8.2.4.10. `gettagConnectionType()`

Declaration:

```
public int gettagConnectionType()
```

Description:

Returns the type of the connection set previously. It can be Mifare Standard, Mifare ultralight or NDEF mode. The detected tag will be read or written using this connection type.

Returns:

int – the type of the connection set previously. Possible values are declared in this class as static variables.

8.2.4.11. `setOffset(int offSet)`

Declaration:

```
public void setOffset(int offSet)
```

Description:

Sets the offset of reading or writing. The Stolpan Host application will read or write offset number of blocks if a tag is detected.

Parameters:

int offSet – new offset value.

8.2.4.12. **settagConnectionType(int tagConnectionType)**

Declaration:

```
public void settagConnectionType(int tagConnectionType)
```

Description:

Sets the connection type. The types are declared in this class as static variables. The Stolpan Host application will use the specific connection type if a tag is detected.

Parameters:

int tagConnectionType – Type of the connection. Possible values are declared in this class as static variables.

9. Package hu.aff.stolpanhost.hsc.screens

9.1. Interface IScreenManager

9.1.1. Declaration

```
public interface IScreenManager
```

Method Summary	
public Screen	getScreen(Class screenRef)
public void	addScreen(Screen screen)
public void	removeScreen(Class screenRef)
public void	displayScreen(Screen screen)
public void	displayScreen(Displayable displayable)
public Screen	getBackScreen()

9.1.2. Description

Interface to the ScreenManager Stolpan Host application component. This component handles the GUI of the whole application. It has a display set by the Stolpan Host application. Every screen has to be registered via this interface in order to be shown. Screens can be registered and removed. The component can be also queried for the last shown screen. Showing a screen is done via the *displayScreen* methods.

9.1.3. Methods

9.1.3.1. getScreen(Class screenRef)

Declaration:

```
public Screen getScreen(Class screenRef)
```

Description:

Returns a screen represented by a class reference.

Parameters:

Class screenRef – the reference of the class to return.

Returns:

Screen – a screen object represented by the screenRef parameter. ScreenRef must be registered previously.

9.1.3.2. addScreen(Screen screen)

Declaration:

```
public void addScreen(Screen screen)
```

Description:

Registers a screen to the screen manager. The screen has to extend the Screen class.

Parameters:

Screen screen – the screen to register in.

9.1.3.3. removeScreen(Class screenRef)

Declaration:

```
public void removeScreen(Class screenRef)
```

Description:

Removes a screen from the screen manager. The screen must be registered earlier.

Parameters:

Class screenRef – Reference of the screen to remove.

9.1.3.4. displayScreen(Screen screen)

Declaration:

```
public void displayScreen(Screen screen)
```

Description:

Displays the given screen identified by the Screen type parameter.

Parameters:

Screen screen – the screen to show.

9.1.3.5. displayScreen(Displayable displayable)

Declaration:

```
public void displayScreen(Displayable displayable)
```

Description:

Displays a displayable object. TPSCs can have other GUI elements to display not just Screen type elements.

Parameters:

Displayable displayable – the displayable object to show.

9.1.3.6. `getBackScreen()`

Declaration:

```
public Screen getBackScreen()
```

Description:

Returns the previously shown screen.

Parameters:**Returns:**

Screen – the previously show screen.

10. Package hu.aff.stolpanhost.hsc.secureelement

10.1. Interface ISecureElementAdapter

10.1.1. Declaration

public interface ISecureElementAdapter extends ITrustedComponent

Field Summary	
public static String	APP_ID

Method Summary	
public byte[]	exchangeApdu(byte[] capdu, String cardletAppId)

10.1.2. Description

Interface to the SecureElementAdapter Stolpan Host component. The component manages the communication process between the Java application and the secure element. The application uses APDU commands for the communication. If a third party service would like to send APDU to its cardlet it have to use the exchangeApdu method. This method has the cAPDU parameter which is the command APDU sending to the cardlet and the cardletAppId parameter which is the application identifier (AID) of the cardlet. The SecureElementAdapter Stolpan Host component first sends a cardlet selection APDU to the secure element. If the response is correct it sends the command APDU to the cardlet then waiting for response APDU. Until the response APDU is not received the Stolpan Host application blocks. If the response APDU arrives the component returns with it and the third party service can handle the response.

10.1.3. Fields

10.1.3.1. APP_ID

Declaration:

public static String APP_ID

Description:

Stolpan Host cardlet application identifier (AID).

10.1.4. Methods

10.1.4.1. exchangeApdu(byte[] capdu, String cardletAppId)

Declaration:

public synchronized byte[] exchangeApdu(byte[] capdu, String cardletAppId) throws IllegalArgumentException, InterruptedException, IOException, IllegalModeException

Description:

Sends a select APDU command to the given cardlet and sends a command APDU after it. Blocks until response APDU is not received.

Parameters:

byte[] capdu – command APDU send to the cardlet

String cardletAppId – application identifier of the purposed cardlet.

Returns:

byte[] – response APDU in byte array.

Throws:

java.lang.IllegalArgumentException – if commandAPDU parameter is null,
if commandAPDU contains a card application selection APDU
if commandAPDU contains a MANAGE CHANNEL command
APDU
if commandAPDU parameter contains a malformed APDU

java.io.InterruptedIOException - can be thrown in any if this Connection object is closed during the exchange operation.

java.io.IOException - is thrown if the operation was not successful because of problems communicating with the card, or if the connection was already closed.

hu.aff.stolpanhost.hsc.exceptions.IllegalModeException – if proximity mode is not set before.

11. Package hu.aff.stolpanhost.hsc.storage

11.1. Interface IStorageAdapter

11.1.1. Declaration

public interface IStorageAdapter

Method Summary	
public int	getFreeSpace(RmsStorageSettings rmsStorageSettings)
public long	getSize(RmsStorageSettings rmsStorageSettings)
public byte []	retrive(FileSystemStorageSettings fileSystemStorageSettings)
public byte []	retrive(RemovableStorageSettings removableStorageSettings)
public byte []	retrive(RmsStorageSettings rmsStorageSettings)
public void	store(FileSystemStorageSettings fileSystemStorageSettings, byte dataToStore)
public void	store(RemovableStorageSettings removableStorageSettings, byte dataToStore)
public void	store(RmsStorageSettings rmsStorageSettings, byte[] dataToStore)
public int	delete(FileSystemStorageSettings fileSystemStorageSettings)
public int	delete(RemovableStorageSettings removableStorageSettings)
public int	delete(RmsStorageSettings rmsStorageSettings)

11.1.2. Description

The interface for the Stolpan Host application Storage Adapter component. It manages save, acces and removal of data from the handsets persistent storage places such as : RMS, FileSystem and Removable Storage.

11.1.3. Methods

11.1.3.1. getFreeSpace(RmsStorageSettings rmsStorageSettings)

Declaration:

```
public int getFreeSpace(RmsStorageSettings rmsStorageSettings)
```

Description:

Returns the amount of additional space (in bytes) available for the given storage.

Parameters:

RmsStorageSettings rmsStorageSettings – settings record about the parameters of the operation.

Returns:

Int – the available space of the given storage place.

11.1.3.2. getSize(RmsStorageSettings rmsStorageSettings)**Declaration:**

```
public int getFreeSpace(RmsStorageSettings rmsStorageSettings)
```

Description:

Returns the amount of the occupied space (in bytes) for the given storage.

Parameters:

RmsStorageSettings rmsStorageSettings – settings record about the parameters of the operation.

Returns:

long - the amount of storage occupied for this store.

11.1.3.3. retrieve(FileSystemStorageSettings fileSystemStorageSettings)**Declaration:**

```
public byte [] retrieve(FileSystemStorageSettings fileSystemStorageSettings)
```

Description:

Retrieves byte array from the File System of the handsets, with fileSystemStorageSettings.

Parameters:

FileSystemStorageSettings fileSystemStorageSettings – a settings record about the parameters of the operation.

Returns:

byte [] – data read by the Stolpan Host framework using given parameters.

11.1.3.4. retrieve(RemovableStorageSettings removableStorageSettings)**Declaration:**

```
public byte [] retrieve(RemovableStorageSettings removableStorageSettings)
```

Description:

Retrieves byte array from the Removable Storage of the handset, with removableStorageSettings.

Parameters:

RemovableStorageSettings removableStorageSettings - settings record about the parameters of the operation.

Returns:

byte [] – data read by the Stolpan Host framework using given parameters.

11.1.3.5. retrieve(RmsStorageSettings rmsStorageSettings)**Declaration:**

```
public byte [] retrieve(RmsStorageSettings rmsStorageSettings) throws Exception
```

Description:

Retrieves byte array from the RMS of the handset, with rmsStorageSettings.

Parameters:

RmsStorageSettings rmsStorageSettings - settings record about the parameters of the operation.

Returns:

byte [] – data read by the Stolpan Host framework using given parameters.

11.1.3.6. store(FileSystemStorageSettings fileSystemStorageSettings, byte dataToStore)**Declaration:**

```
public void store(FileSystemStorageSettings fileSystemStorageSettings, byte dataToStore)
```

Description:

Stores the byte stream getting from dataToStore parameter to the filesystem of the handset using parameters from fileSystemStorageSettings record.

Parameters:

FileSystemStorageSettings fileSystemStorageSettings – settings record about the parameters of the operation.

byte dataToStore – data to write to the storage.

11.1.3.7. store(RemovableStorageSettings removableStorageSettings, byte dataToStore)**Declaration:**

```
public void store(RemovableStorageSettings removableStorageSettings, byte dataToStore)
```

Description:

Stores the byte stream getting from dataToStore parameter to the removable storage of the handset using parameters from removableStorageSettings record.

Parameters:

RemovableStorageSettings removableStorageSettings - settings record about the parameters of the operation.

byte dataToStore – data to write to the storage.

11.1.3.8. store(RmsStorageSettings rmsStorageSettings, byte[] dataToStore)

Declaration:

public void store(RmsStorageSettings rmsStorageSettings, byte[] dataToStore) throws Exception

Description:

Stores the byte stream getting from dataToStore parameter to the record management system of the handset using parameters from rmsStorageSettings record.

Parameters:

RmsStorageSettings rmsStorageSettings - settings record about the parameters of the operation

byte dataToStore – data to write to the storage.

11.1.3.9. delete(FileSystemStorageSettings fileSystemStorageSettings)

Declaration:

public int delete(FileSystemStorageSettings fileSystemStorageSettings) throws DataNotFoundException

Description:

Deletes a Dataset from the FileSystem Storage. Identified by the fileSystemStorageSettings object.

Parameters:

FileSystemStorageSettings fileSystemStorageSettings – settings record about the parameters of the operation.

Returns:

int – value 1 if the delete was succesfull else throws exception.

Throws:

hu.aff.stolpanhost.hsc.exceptions.DataNotFoundException – If the delete operation was not succesfull.

11.1.3.10. delete(RemovableStorageSettings removableStorageSettings)

Declaration:

public int delete(RemovableStorageSettings removableStorageSettings)

Description:

Deletes a Dataset Identified by the fileSystemStorageSettings object from the Removable Storage Settings.

Parameters:

RemovableStorageSettings removableStorageSettings – settings record about the parameters of the operation.

Returns:

int – value 1 if the delete was successful, else throws exception.

Throws:

hu.aff.stolpanhost.hsc.exceptions.DataNotFoundException – If the delete operation was not successful.

11.1.3.11. delete(RmsStorageSettings rmsStorageSettings)**Declaration:**

public int delete(RmsStorageSettings rmsStorageSettings) throws Exception.

Description:

Deletes a Dataset from Removable Storage Settings. Identified by the fileSystemStorageSettings object.

Parameters:

RmsStorageSettings rmsStorageSettings – settings record about the parameters of the operation.

Returns:

int – value 1 if the delete was successful else throws exception.

Throws:

hu.aff.stolpanhost.hsc.exceptions.DataNotFoundException – If the delete operation was not successful

12. Package hu.aff.stolpanhost.util

12.1. Interface Persistent

12.1.1. Declaration

public interface Persistent

Method Summary	
byte[]	persist()
void	resurrect(InputStream in)

12.1.2. Description

A simple interface for building persistent objects on platforms where serialization is not available. Classes to serialize must implement this interface. Serialized objects are used to store data i.e. using Storage Adapter Stolpan host component.

12.1.3. Methods

12.1.3.1. persist()

Declaration:

byte[] persist() throws java.io.IOException

Description:

Called to serialize an object to a byte array. Third party services must implement this method referring to their classes.

Returns:

byte[] – serialized object representing in a byte array.

Throws:

java.io.IOException – if serialization failes.

12.1.3.2. resurrect(InputStream in)

Declaration:

void resurrect(InputStream in) throws IOException

Description:

Called to deserialize a persistent object from an InputStream. Third party services must implement this method referring to their classes.

Parameters:

InputStream in – input stream object which points to a byte array containing a serialized object.

Throws:

java.io.IOException – if deserialization fails

12.2. Class ByteUtil

12.2.1. Declaration

```
public class ByteUtil
```

Method Summary	
public static String	byteArrayToString(byte[] array)
public static byte[]	stringToByteArray(String string)
public static boolean	byteArraysMatch(byte[] array1, byte[] array2)
public static byte[]	getBytes(int num)
public static String	intToAscii(int num)

12.2.2. Description

An utility class for performing various byte handling functions.

12.2.3. Methods

12.2.3.1. byteArrayToString(byte[] array)

Declaration:

```
public static String byteArrayToString(byte[] array)
```

Description:

Convert a byte array to a String of hexadecimal digits.

Parameters:

byte[] array – byte arra to convert.

Returns:

String - the resulting String of hexadecimal digits.

12.2.3.2. stringToByteArray(String string)

Declaration:

```
public static byte[] stringToByteArray(String string)
```

Description:

Convert a String of hexadecimal digits to a byte array.

Parameters:

String string - the String of hexadecimal digits to convert.

Returns:

byte[] - the resulting array of bytes.

12.2.3.3. byteArraysMatch(byte[] array1, byte[] array2)**Declaration:**

```
public static boolean byteArraysMatch(byte[] array1, byte[] array2)
```

Description:

Return true, if the two byte arrays contain exactly the same bytes.

Parameters:

byte[] array1 – first array to check.

byte[] array2 – second array to check.

Returns:

boolean – return true, if the two byte arrays contain exactly the same bytes.

12.2.3.4. getBytes(int num)**Declaration:**

```
public static byte[] getBytes(int num)
```

Description:

Convert an int to a four-byte-array using `DataOutputStream`.

Parameters:

int num – the number to convert.

Returns:

byte[] - byte array represents the int parameter.

12.2.3.5. intToAscii(int num)**Declaration:**

```
public static String intToAscii(int num)
```

Description:

Converts an integer to its ASCII value. ASCII number values are in the 30-39 interval.

Parameters:

int num – number to convert to its ASCII value.

Returns:

String – the ASCII value of the number in string format.

12.3. Class Sleeper

12.3.1. Declaration

```
public class Sleeper
```

Method Summary	
public static void	mySleep(int millisec)

12.3.2. Description

This utility class has only one method. Its task is to suspend a running thread.

12.3.3. Methods

12.3.3.1. mySleep(int millisec)

Declaration:

```
public static void mySleep(int millisec)
```

Description:

Sleep for the given number of millisecs.

Parameters:

int millisec – the time value in millisecondum to sleep the thread.

12.4. Class ResourceBundle

12.4.1. Declaration

```
public class ResourceBundle
```

Field Summary	
protected Hashtable	resources

Method Summary	
public static String	getString(String group, String key)

12.4.2. Description

Every TPSC may have texts in different languages. The ResourceBundle class is used to build text repositories and set up a context to use texts in different languages depending on the language settings of the Stolpan Host application.

TPSCs have to create their own language repository classes. The name of a language repository class has the following format:

"Optional family class name" _Locale ID _Country ID

Locale ID identifies the language using international identifier. For example English language has the identifier "en", german language has the identifier "de".

Country ID identifies the country where the language is used. There may be differences between translations in different countries.

I.e: Test_en_EN

A resource bundle class of a TPSC has to extend the ResourceBundle class. TPSCs have to use the resources Hashtable object to store their texts to the repository. Typically TPSCs use the resources.put methods in the constructor of the language repository class. The resources.put method has two parameters: the first is the key of the text and the second is the text translation of the given language.

If the TPSC would like to use the text it has to use the getString static method. The getString method identifies the repository class of the TPSC using the group parameter which is the path of the class (com.mycompany.myproject.resources.test). The key parameter of the getString method identifies the text itself. The text language will be based on the language settings of the Stolpan Host application.

12.4.3. Fields

12.4.3.1. resources

Declaration:

protected Hashtable resources

Description:

Repository for the texts of a language settings. Using resources.put method TPSC can store the translation of a text in a language. First parameter of put method is the key of the text the second parameter is the translation of it to the given language.

12.4.4. Methods

12.4.4.1. getString

Declaration:

public static String getString(String group, String key)

Description:

Returns a translation of a text belonging to a TPSC.

Parameters:

String group – identifies the resource family. Important: Do not use the full path of the class. Only have to identify the family of the resource classes. I.e if the resource class have the following path: com.mycompany.myproject.resources.test_en_EN you have to specify the com.mycompany.myproject.resources.test String in this parameter.

String key – identifies the text itself. Every text has a key.

Returns:

Returns the translation of the text identified by the parameters depends on the settings of the Stolpan Host application.