

Panel Service v1 Developer's Guide

SDK/J Authentication Package Version:1.0



Copyright © 2008 Ricoh Co., Ltd.

Terms of Use and Trademarks

1. The contents of this specification may be changed without notice in the future.
2. The copying, reproducing, changing, quoting, reprinting, or distributing a part/all of this book are prohibited.
3. We will not be held responsible for any of our customer's losses, disadvantages, or demands from a third party on using this book.
4. Trademarks
 - Ethernet® is a registered trademark of Xerox Corporation.
 - PostScript® and Acrobat® are registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.
 - Microsoft® and Windows® are registered trademarks of Microsoft Corporation in the United States and other countries.
 - UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Ltd.
 - Red Hat is a registered trademark of Red Hat, Inc. in the United States and other countries.
 - Linux is a registered trademark of Linus Torvalds in the United States and other countries, or both.
 - Java, JavaCard, JVM(CVM), and CDC are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.
 - Eclipse is a trademark or a registered trademark of eclipse.org in the United States and other countries.
 - OSGi(TM) is a trademark or a registered trademark of The Open Service Gateway Initiative in the United States and other countries.
 - Apache is a registered trademark of The Apache Software Foundation in the United States, other countries, or both.

Other product names used herein are for identification purposes only and might be trademarks of their respective companies. We disclaim any and all rights in those marks.

Contents

1. Introduction	2
1.1. Target Readers.....	2
1.2. Software Requirements	2
1.3. Restrictions	2
2. Overview	3
2.1. Ccm class	5
2.2. CcmService class	5
2.3. CcmRequestHandler interface	5
2.4. CcmRequest interface	5
3. Panel transition diagram	6
4. Application Development	7
4.1. Obtaining an instance of the CcmService class	7
4.2. Implementing the requestLogin() method and the requestLogout() method	7
4.3. Implementing the CcmRequestHandler interface and registering it with the CcmService class	8
5. Application Installation	10
5.1. DALP file settings	10
5.2. Installing the application	10
Change History	11

1. Introduction

This document describes how to use the Panel Service.

The Panel Service is a software framework necessary to use the user authentication management function of an MFP/LP in an SDK Type-J (hereinafter referred to as SDK/J) application.

1.1. Target Readers

This document is intended for the application developers who are experienced in SDK/J application development and familiar with the access role function of an MFP/LP.

1.2. Software Requirements

Use of the Panel Service requires an operation environment where:

- Panel Service is enabled.

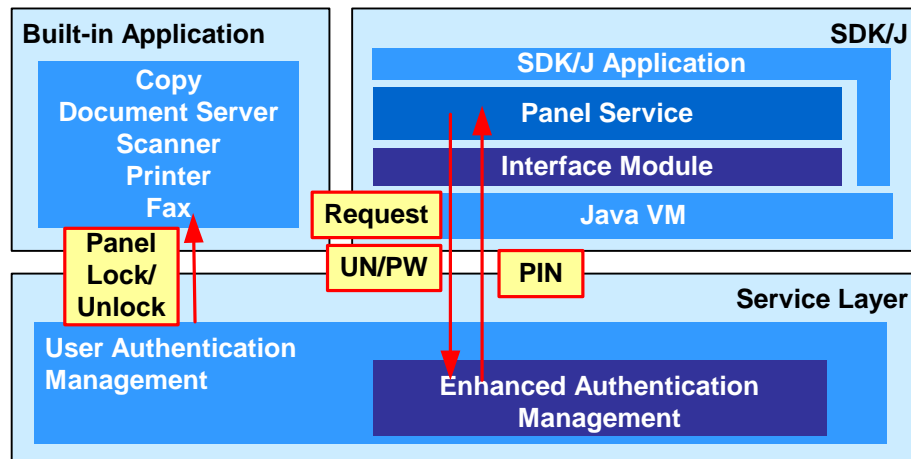
* About the setting of Panel Service, see “SDK/J Authentication Package Settings Guide”.

1.3. Restrictions

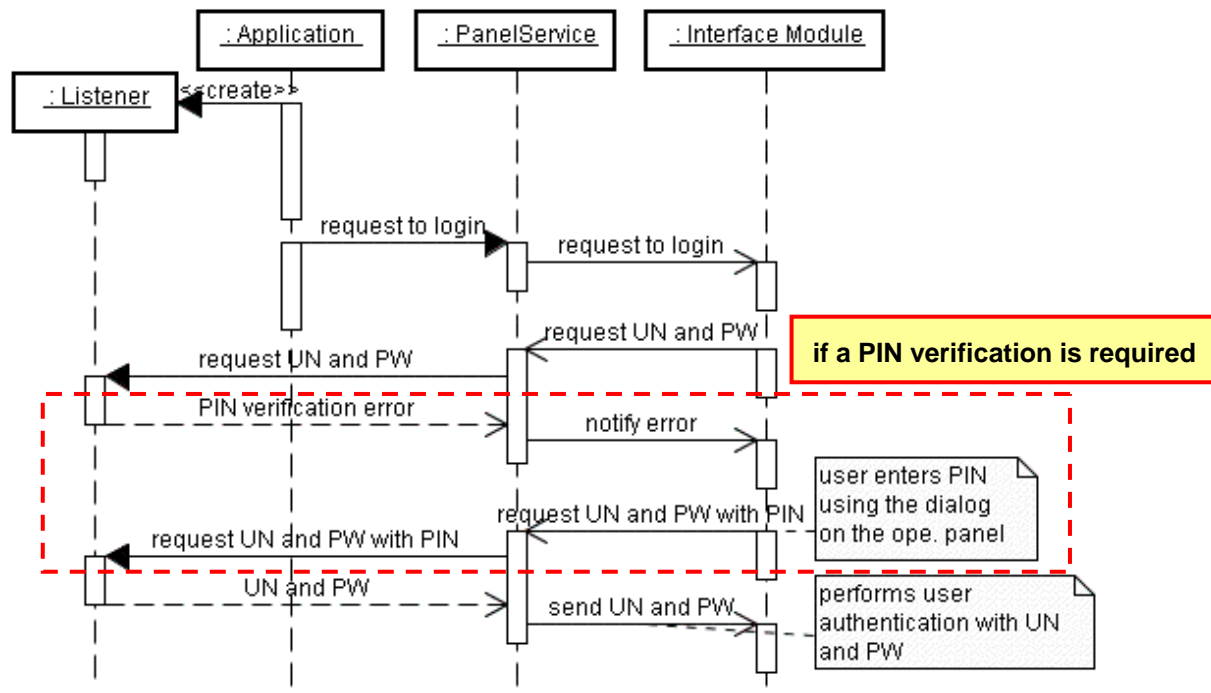
- The Panel Service can not be used by multiple applications at one time. That is, at any time, only one application can use the Panel Service. So when more than one application that uses the Panel Service is active at the same time, only the application that started first can use the Panel Service.
- The handle() method in CcmRequestHandler must be processed within 30 seconds, otherwise a login request will be failed due to the specification of the user authentication module.
- The screen images for user authentication can not be customized using the Panel Service.
- The available characters and the length of the login user name or password depend on the access role specification of the target device. The PIN must be 1 to 8 digits.
- The login request from the Panel Service is valid when the authentication request dialog is displayed on the operation panel of the target device.
- To use the logout feature of the Panel Service, the System firmware of the target device has to support the logout feature with the enhanced authentication management. For details, please contact RiDP.

2. Overview

The Panel Service is a software framework that allows SDK/J applications to use the user authentication management of MFP/LP with a login user name and a login password (hereinafter referred to as UN and PW). The following picture shows how the Panel Service and its related software components are deployed. The Panel Service communicates with the user authentication module that resides in the System firmware through the Interface Module provided with Authentication Package Bundle and lock/unlock the panel.

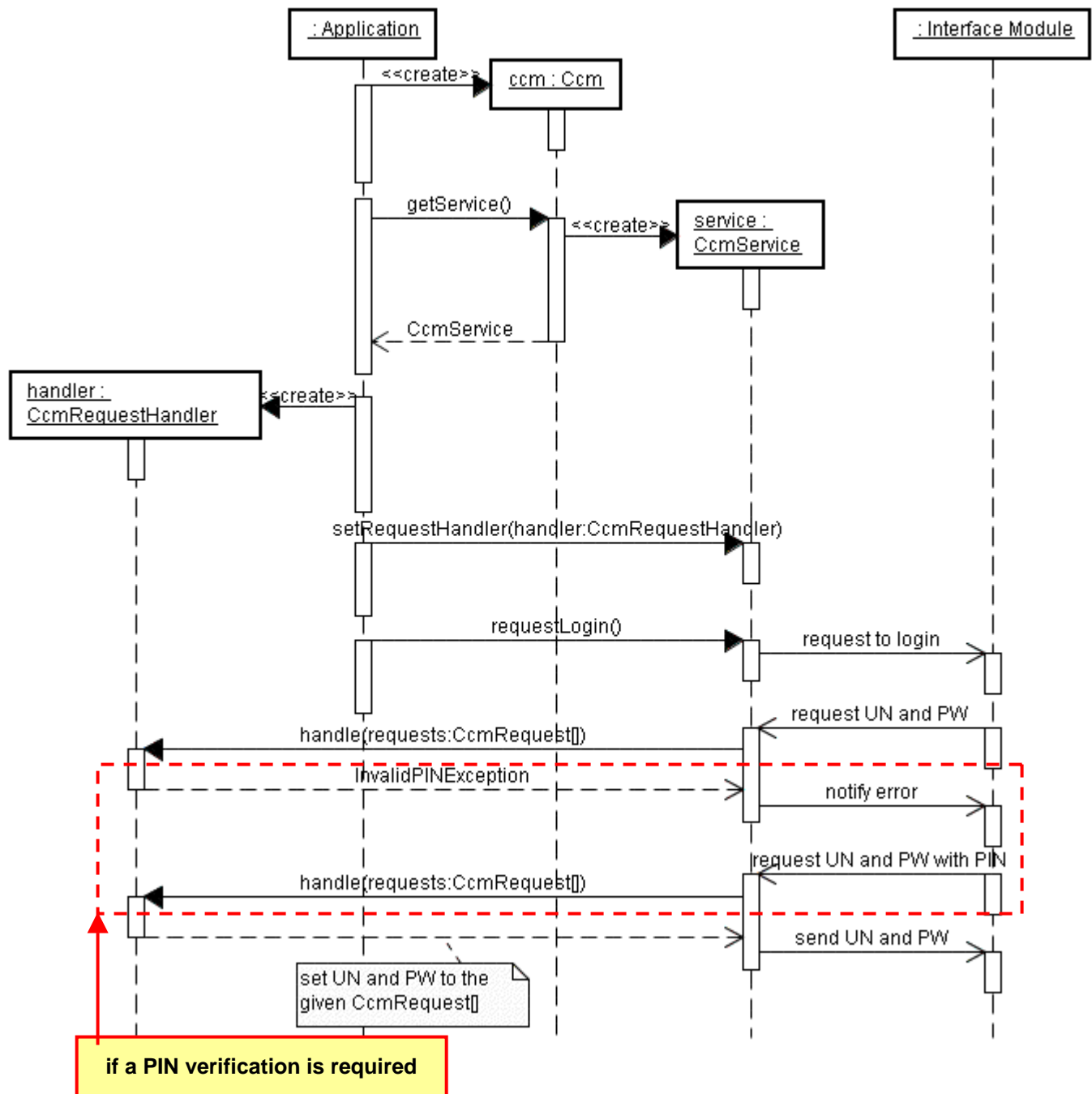


The sequence diagram below shows the operation sequence of the Panel Service. The application sends a login request to the Panel Service at the appropriate time (for example, when a smart card for the user authentication is detected). Then the Panel Service forwards the login request to the interface module. On receiving the login request, the interface module makes a request for UN and PW. All these requests are handled by the listener that has been registered by the application in advance. If a PIN verification is necessary to obtain UN and PW, the listener notifies the interface module of the PIN verification error and then the user authentication module displays the PIN input dialog that prompts the user to input the PIN.



The following sequence diagram shows the actual operation sequence and what methods are used in it. The application obtains a CcmService object by using a Ccm object, and then makes a login request by using the requestLogin() method. A CcmRequest request such as a request for UN and PW or PIN verification is handled by the handle(CcmRequest[]) method of a concrete CcmRequestHandler handler.

If a PIN verification is required to obtain UN and PW, an PIN input dialog will be displayed on the panel by throwing InvalidPINException in the handle(CcmRequest[]) method.



The major classes used in the Panel Service are as follows.

For more details, see the Javadoc.

2.1. Ccm class

The Ccm class is the front class for the application.

Its getService() method obtains an instance of the CcmService class.

2.2. CcmService class

The CcmService class notifies the user authentication module of a login and a login cancel / logout by using its requestLogin() method and requestLogout() method respectively.

Also it handles requests made by the user authentication module by using the concrete CcmRequestHandler handler that it has registered with itself by using the setRequestHandler() method.

Note: A maximum of one such object can be registered with this class.

In other words, this class can not register more than one such object with itself.

2.3. CcmRequestHandler interface

This is the interface that defines the handle() method that handles requests made by the user authentication module.

This interface must be implemented by the Panel Service application developer.

2.4. CcmRequest interface

This is the interface that indicates the request was made by the user authentication module.

In the Panel Service, this interface is implemented by:

- The Name class that represents UN.
- The Password class that represents PW.
- The VerifyPIN class that represents a PIN verification request.

4. Application Development

This chapter explains how to implement an application that uses the Panel Service.

Use **panelservice.jar** to develop a Panel Service application.

Details on SDK/J application development and exception handling are not described here.

4.1. Obtaining an instance of the CcmService class

Obtain an instance of the CcmService class by instantiating the Ccm class and using the getService() method of it.

```
.
.
/** Obtain an instance of the CcmService class. */
CcmService ccmService = new Ccm().getService();
.
.
```

4.2. Implementing the requestLogin() method and the requestLogout() method

Implement the requestLogin() method and the requestLogout() method appropriately.

These methods notify the user authentication module of a login and a login cancel / logout respectively; they should be implemented as they are called properly.

In the sample code below, the requestLogin() method is called when a card is detected, and the requestLogout() method is called when the card is removed from the card reader device.

```
.
.
CardManager cardManager = new CardManager();

cardManager.addCardListener(
    new CardEventListener() {
        /** Notify the user authentication module of a login
         * when a card is detected. */
        public void inserted(CardEvent e) {
            ccmService.requestLogin();
        }

        /** Notify the user authentication module of a login cancel / logout
         * when a card is removed from the card reader device. */
        public void removed(CardEvent e) {
            ccmService.requestLogout();
        }
    }
);
.
.
```

4.3. Implementing the CcmRequestHandler interface and registering it with the CcmService class

Implement the CcmRequestHandler interface that handles requests made by the user authentication module. The handling of each CcmRequest such as Name, Password, or VerifyPIN, is implemented by the handle() method. In the method, CcmRequests must be categorized at first, then handled by each.

If a PIN verification-related error has occurred during a login process, the handle() method should throw the InvalidPINException exception and the Panel Service should notify the user authentication module of the error.

```

CcmRequestHandler myHandler = new CcmRequestHandler() {
    /** Implement the handle() method. */
    public void handle(CcmRequest[] requests)
        throws UnsupportedOperationException, InvalidPINException, IOException {
        Name name = null;
        Password password = null;
        VerifyPIN pin = null;

        /** Categorize requests appropriately. */
        for (int i = 0; i < requests.length; i++) {
            if (requests[i] instanceof Name) {
                name = (Name) requests[i];
            } else if (requests[i] instanceof Password) {
                password = (Password) requests[i];
            } else if (requests[i] instanceof VerifyPIN) {
                pin = (VerifyPIN) requests[i];
            } else {
                /** Throw an exception when the request is improper. */
                throw new UnsupportedOperationException();
            }
        }

        /** Handle each request. */
        try {
            if (pin != null) {
                byte[] pinBuffer = pin.getPIN();
                /** Perform a PIN verification. */
                cardService.verifyPIN(pinBuffer);
            }
            if (name != null) {
                /** Obtain the login user name. */
                byte [] nameBytes = cardService.getUser();
                name.setName(nameBytes);
            }
            if (password != null) {
                /** Obtain the login password. */
                byte [] passBytes = cardService.getPassword();
                password.setPassword(passBytes);
            }
        } catch (CHVException e) {
            /** Throw an exception when an error has occurred in the PIN verification. */
            throw new InvalidPINException();
        }
    }
};

```

Register the implemented CcmrequestHandler handler with a CcmService object by using the setRequestHandler() method. Requests that are made by the user authentication module are handled by this registered CcmRequestHandler handler.

```
.  
.br/>/** Register the CcmRequestHandler handler with the CcmService class. */  
ccmService.setRequestHandler(myHandler);  
.br/>.
```

Note: The CcmService class can register a maximum of one CcmRequestHandler handler with itself. Therefore, when the CcmService class already has a CcmRequestHandler handler, the CcmService should throw the AlreadySetException exception.

The registered CcmRequestHandler handler must be removed by the removeRequestHandler() method when the application is stopped.

```
.  
.br/>/** Remove the registered CcmRequestHandler handler. */  
ccmService.removeRequestHandler(myHandler);  
.br/>.
```

5. Application Installation

This chapter describes how to install an SDK/J application that uses the Panel Service.

5.1. DALP file settings

About DALP format, see the SDK/J Developer's Guide.

Since the Panel Service jar file (panelservice.jar) is deployed in SDK/J SD card, jar file description of the panelservice.jar is unnecessary.

5.2. Installing the application

The installation procedure is the same as a normal SDK/J application.

For details, see the User's Guide of the SDK/J.

Note: When the option-jar tag is used in the DALP file, it is needed to reboot the target device after the installation.

Note: Since the Panel Service works in conjunction with the MFP/LP firmware-resident user authentication module, an SDK/J application using the Panel Service does not work on an emulator.

Note: After the installation, You must configure the authentication management settings of the target MFP/LP appropriately. For details, see "Panel Service v1 User's Guide".

Change History

Ver. 1.0	SDK/J v4 or later version based on SDK/J AP v1.0 option package for SDK/J v2. Update software requirements Implementation-Version:1.1-1.1
----------	---