2005－GN－55（15）
2005／3／18

An Approach to Integrate Smartcards Functionalities into Various Web Applications

Michel FAURE　Junko HASHIMOTO　Tomohiro KOKOGAWA

Shinichi HIRATA　Sei IJUIN

NTT Service Integration Laboratories

E-mail:　{michel.faure, hashimoto.junko, kokogawa.tomohiro, hirata.shinichi, ijuin.sei}@lab.ntt.co.jp

Abstract: Services that makes use of Smartcards are spreading. However, because of lack of data interfacing or security problems, few Web applications use Smartcards to its full potential. We propose in this paper an architecture that permits easy construction or upgrade of standard Web applications so that it makes use of Smartcard added functionalities. A Smartcard services support library intended for Web applications developers was constructed and we report the results of prototypes made based on this architecture.

.

Keywords　Smartcards,　MVC Model, Security

1. Introduction

In the recent years, smartcards have been adopted in various fields where high security and high performance are required [1]. By offering strong anti-counterfeiting measures and thus leveraging security found on previous magnetic stripe technology cards, smartcards have become a convenient device of choice for different institutions such as banking, administration [2] and transportation [3].

In parallel, web applications have also been wildly implemented at various scales in these same sectors. Transactions processing systems relying on web applications and associated network enabled web browsers terminals can now be deployed easily and at low-cost.

However, the rising number of internet security threats put most of these web applications at risk. Smartcards have been an ideal candidate for that matter, not only to re-enforce the security level but also to easily personalize the delivered web services experience for each cardholder.

The goal of this paper is to propose an architecture that allows easy mounting and use of Smartcards functionalities into web applications. Several components like a server-side middleware smartcard library and client-side components are used to facilitate this integration. Using these components, it is easy to upgrade an existing web application so that it makes use of whole new smartcards possibilities.

A web application was built based on this architecture, the results are discussed. For commodity and in order to prototype a system, we used the NICE card management platform and ELWISE [5] card developed by NTT Laboratories as an underlying base platform for our architecture.

2. Smartcard services construction problem

2.1. Context and target

As fore mentioned, smartcard functionalities are an asset whenever generic Web applications handles sensitive information like customer data must be made.

For our study, we will focus on a concrete example: a corporate Web portal that manages employee cards. Services related are expected to make best use smartcard functionalities and provide the following online and offline services:

- Intranet access to the corporate Web portal
- Single sign on for multiple services use
- Conference room booking and library material rental management
- Private information management (profile, working hours, phone book, etc…)
- Offline use: micro-payment, room access

2.2. Smartcard services system construction

Specific in-house systems are more and more

upgraded in the form of Web applications. The MVC (Model View Controller) design pattern is mostly used as it is well adapted to that case. Data and business logic is often described with EJB, the view is realized with JSP/HTML pages and the service transactions are executed with controller servlets.

On the other hand, considerations are different for smartcards as specific management concerning smartcard's contained data is necessary. Therefore, when existing Web applications wants to make use of smartcards functionalities it is necessary to:

- Provide mapping a scheme between Web services and on-card applications.
- Make each smartcard service related data and Web service data consistent for process.
- Provide a session oriented scheme that encompasses all service processes.

It appears that smartcard and Web applications integration is not a simple task.

3. Approach

We chose to use the well accepted and MVC compliant technology to build such dynamic Web services: the JSP/Servlets/JDBC technology.

3.1. Requirements

Our architecture must satisfy the following:

- A library of functions must be offered for Smartcard related operations.
- Processing sequence must be flexible and benefit from smartcard security to allow secure service use.
- Web pages construction and design should remain easy for web designers.

Our architecture adapts the MVC design pattern concepts to smartcards concepts.

3.2. Adapting the MVC model

When Smartcards functionalities are mounted on top of the MVC model, the Model, View and Controller entities becomes the following:

Model : the data stored into the card and the data stored into the application server database.

View : Web application's GUI

Controller : As in traditional MVC based Web applications, the Controller is a servlet.

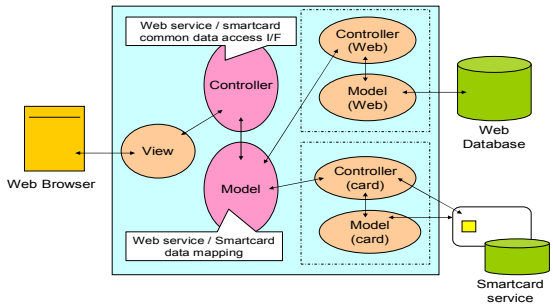

Figure 1. MVC model applied to smartcards

3.3. Smartcard data management

Web applications generally accesses data located in remote databases through SQL queries or Java Beans. On the other hand, the data contained in smartcards applications, JICSAP-based[6] in our case, is accessed via APDU commands (Application Protocol Data Units). The way to access these smartcard data is dependent on the type of smartcard considered and the specifications of applications loaded onto it. Thus, a change in smartcard specifications leads to incompatibility problems: the APDU that get on-card data needs to be modified.

We propose a method to make a one-to-one mapping of the data located on the card with the data located in the service server database. It is thus possible to reduce and localize the code changes in a Web application based on our architecture.

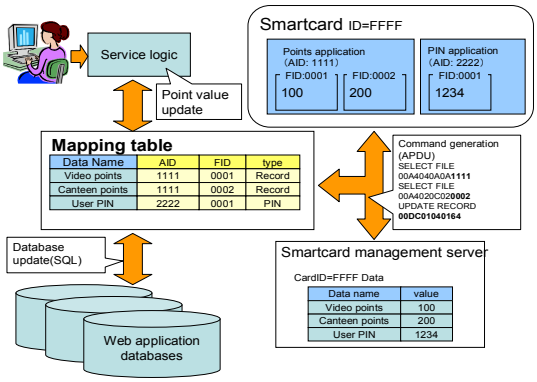

Figure 2. Service mapping with smartcard

We also propose an XML based scheme to collect or update via APDU commands any data located on the card in a flexible way.

3.4. Session management for smartcards services

Smartcards are well-known for being tamper resistant devices; sensitive information like secret keys data is securely confined into the card's memory. A web application that needs secure session management makes the best use of this feature.

A service needs to create and maintain a secure and persistent connection over several contents, from login to logout. Although the smartcard holder authentication credentials can be checked at login time, a mechanism is needed to maintain the authentication state: most often the HTTP session ID is used. However, there is no direct link between the card authentication credentials and the session id. A bridge between the card management system and the Web services security mechanisms is thus necessary. We propose to implement such bridge with a session oriented management table.

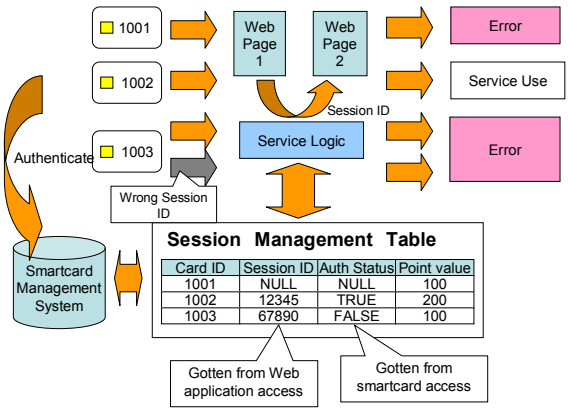

Figure 3. Session management

4. Implementation

4.1. Overview

In order to confirm the utility of our research, we built a prototype system based on our architecture. We tried to identify the smartcard functionalities that could be of use for Web designers and defined accordingly smartcard service mapping databases. We also chose the card applications internals to be compatible and defined card data access and update mechanisms. Then, we developed individual functions in form of a JSP custom tag library to support Smartcard services from within JSP pages. To interface smartcards from the local terminal browser, an ActiveX control was chosen. This library's aims to become a tool to build Web applications that, although very different in shapes and contents, use this very same library to deliver smartcard enabled services.

Our developing environment was the following:

- JRUN4 and IIS servers
- NICE card management system/ELWISE card

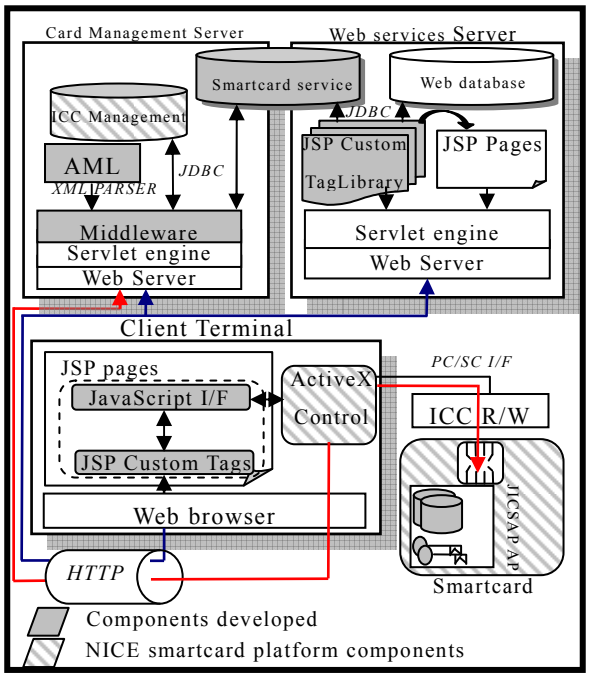

Figure 4. Prototype system overview

The different elements depicted in the above figure fall into one category of the MVC model described in figure 1.

Table 1: MVC model correspondence

Controller	Custom tags	Web	Custom tags
		Card	Middleware/AML
Model	Smartcard service database	Web	Web database
		Card	On card data
View	JSP		

4.1.1. Integration with the NICE Smartcard platform

Although our system can probably cohabit with any kind of smartcard management system, for commodity and in order to develop a prototype, we used the NICE card management platform and ELWISE card developed by NTT Laboratories.

4.1.2. Smartcard application architecture

Among the variety of smartcard specifications, we chose to use the JICSAP specifications, which are the Japanese Smartcard specifications counterpart of the well-known ISO 7816 smartcard specifications. It allows definition of file structure within an application and defines data access security mechanisms.

4.2. Smartcard services support library

JSP custom tags encapsulate complex transactions logic into simple and accessible components. Once defined in a tag library, it is possible to use them in JSP pages like standard HTML tags. Customs tags can manipulate dynamically the page contents. Custom tags encourage division of labor between Web designers and developers and factors recurring processes for reuse across projects. This smartcard services library aims to let Web designers use smartcards mechanisms as described in figure 5.

4.2.1. JSP custom tag library

The library custom tags are partitioned according to different purposes:

- Authentication
- Smartcard applications management
- Server and on-card data management

The tags defined here below are customized via attributes, passed from calling page, that can be hard-coded or resulting of JSP code execution (session variables, HTML form data, etc…).

Table 2. Custom tags definition

Type		Tag Name	Description
Authentication		**Login**(*)	Register browser session ID with the login time. PIN verification and PK authentication is possible.
		Logout	Reset the authentication status and date.
		SecureContents	Display the tag's body contents if and only if this user has logged. Verify for session timeout.
		PIN Change(*)	Change card's PIN locally.
		PIN Init(*)	Init card's PIN directly from the server.
Smartcard application management		**AP download**(*)	Remotely download a new card application. This card application is associated to a service, and the card application data are mapped to the service data.
		AP Deletion(*)	Remotely delete a card application
Data Management	Smartcard	**Personalize** (*)	Personalize on-card data using with the data contained in service related databases.
	Server data management	**Backup**(*)	Get on-card data and save the results in the service databases.
		Update(*)	Re-personalize the on-card data with the data contained in the service related databases.
		GetDBData	Retrieve data from the service related databases.
		UpdateDBdata	Update data in the service related databases.

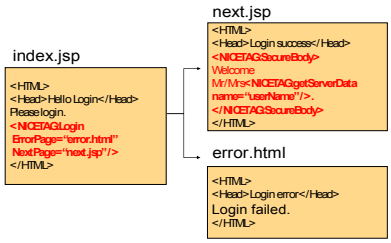

Figure 5. Custom tag use in JSP pages

The tags signaled with (*) indicates that actually two tags are used: a body tag to place in the page's body and a header tag to place in the header. The header generates JavaScript code to interface the local ActiveX control that communicates with the card applications.

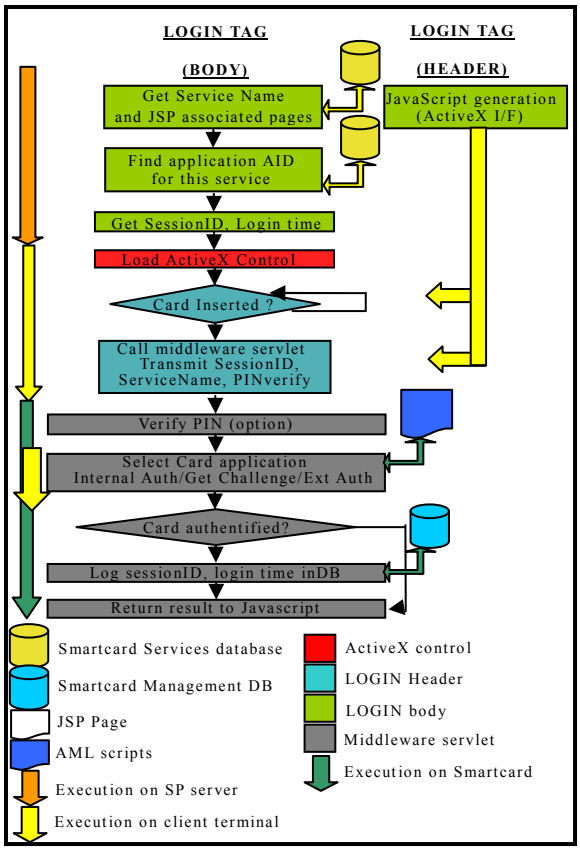

Figure 6. LOGIN tag execution flow

4.2.2. Custom tag execution flow

On figure 6, we detail the behavior of the LOGIN custom tags.

4.2.3. Smartcard interface control

To access smartcard locally (ChangePIN) or remotely from the middleware servlet (BackupAPData, etc…) we need a mechanism to access locally or remotely on-card data. We choose to use an ActiveX control that aims to communicate with card applications and offer functions entry points accessible with JavaScript to perform:

- Multipurpose APDU Transmission
- PIN Verification/Update
- Connection to the smartcard server servlet

The results returned from this control are handled with Javascript event callback functions:

- Card APDU Response event
- Smartcard Server servlet response event

The Javascript code to interface the control is generated by [Header] custom tags (ex: Login).

On the server side, the Middleware servlet located on the Smartcard Management server performs remote card access using AML files. AML scripts are resource files written according to the XML syntax [7]. For each card application, and thus each Web service, two AML files are used:

- An APDU dictionary file
- A service-oriented file that refers to the APDU dictionary file and defines execution processes for smartcard service (data personalization /backup).

This mechanism is transparent for Web designers.

4.3. Smartcard services databases

4.3.1. Definition

The smartcard service database described in the figure 4 is actually made of the following tables:

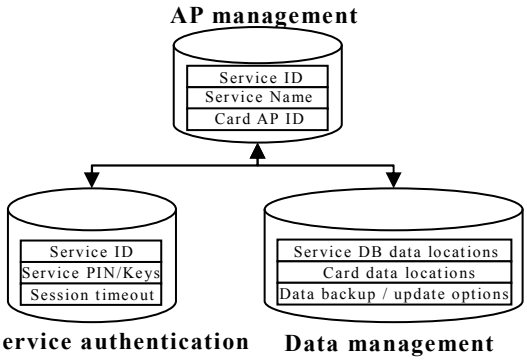

Figure 7. Smartcard service oriented tables

The AP Management performs a one-to-one mapping between a Web service and a smartcard application.

The service authentication table describes the smartcard featured security options to use for each service.

The Data management table defines the data referenced by a service and maps these data's location with on-card embedded application's data.

4.3.2. Service mapping

The AP management database assigns each on-card JICSAP application, represented by its AID identifier, to one Web service.

4.3.3. Service authentication

The service authentication database defines security mechanisms for service login or data access during a session. Smartcard provided security mechanisms include PIN verification, PK authentication and Secure Messaging.

4.3.4. Service data management

The data mapping in each on-card application's file record is defined in the service Web server databases. Access to this data is protected by Mutual Authentication and Secure Messaging. Only the middleware servlet has the right to access these card applications' data. The way to access is described with AML scripts.

4.4. Session management

The session ID returned by the middleware servlet is used as it is. The service authentication

table 4.3 is used for authentication. Security level for each service can be parameterized:

- Authentication necessity
- Secure messaging necessity

Authorization level check is performed automatically by the tag library and spares the Web designers from these session management hurdles.

5. Evaluation

5.1. Web application prototype : A corporate employee portal

This prototype is a corporate Web portal. Employees can connect to the in-house system with their smartcards, access to different services like room reservation, material lending. Offline use support is also provided (canteen application).

For prototyping and tests, we used Dreamweaver MX.

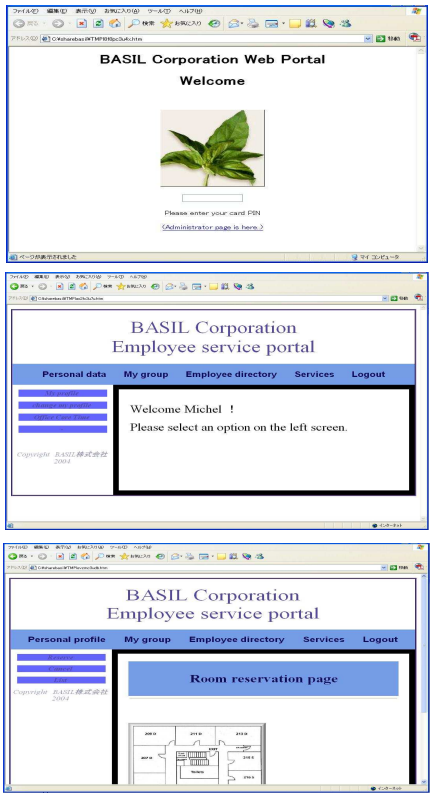

Figure 8. Corporate Web portal

5.2. Discussion

We dress the results of our prototype in the table here below.

Table 3. Prototype evaluation results

	Corporate Web portal
Number of additional service tables	3
Number of JSP pages	12
Number of custom tags used	30
Average number of custom tags per page	2.5

According to these results, we believe it is possible to deploy smartcard enabled Web applications with little JSP code overhead. Other prototypes development is on-going at the time we write this document and will be presented later. The future results will determine if further smartcard service tags have to be deployed for better functionality.

6. Summary

The architecture we developed in our research try to make use of all smartcard functionalities when using Web applications. The prototypes based on our smartcard service library show that with little effort Web applications and smartcards synergy in various contexts is possible.

References

[1] “IC Card Souran 2003-2004”, Seamedia, 2004 (In Japanese).

[2] http://www.soumu.go.jp/c-gyousei/daityo/ (In Japanese)

[3] “Suica”, http://www.jreast.co.jp/e/development/activities/satisfaction/ticketless.html

[4] R. Toji, Y. Wada, S. Hirata and K. Suzuki, “A Network-based Platform for Multi-application Smart Cards”, Proc. 5th International Enterprise Distributed Object Computing Conference (EDOC 2001), IEEE Computer Society Press, 2001.

[5] M. Yoshizawa, H. Unno, T. Fukuzawa, and H. Ban, “ELWISE – A super multi-purpose smart card”, NTT Review, Vol. 16, No. 1, pp.23-27, 2002.

[6] “JICSAP”, http://www.jicsap.com/

[7] Junko HASHIMOTO, Katsuhiko SUZUKI, Shinichi HIRATA, “Definition on smart card service process sequence using XML”, IEICE KBSE2002-21, pp 1-6, 12/2002.