Palette UI | Browser iC Function | i-appli Extensions

Technology Reports

Application Functions for Winter/Spring 2011-2012 Models

*The increasingly diverse and complex nature of functions and services provided by mobile terminals may cause users to miss out on ones that they would like to use since navigating to them is not always easy even if they are installed on the terminal. The NTT DOCOMO winter/spring 2011-2012 models come equipped with the Palette UI function, in-browser FeliCa®*1 access function, and i-appli function extensions to shorten the path to functions and services and improve functionality.*

Communication Device Development Department

Taeko Yamaki
Tetsuro Susa
Hisato Ogawa
Ryo Nakajima
Takashi Yoshikawa
Keiichi Murakami

1. Introduction

The rise of the smartphone and the growth of an advanced and diversified mobile market in recent years have presented the user with a flood of services, products, and information. This situation has generated a need for information-access tools to improve terminal operability and life-support tools as well. At the same time, the provision of a wide variety of functions, applications, and services has created an issues in that users may miss out on functions or services that they would like to use since they are not always easy to reach on the terminal [1].

In response to this issue, we developed a User Interface (UI) called Palette UI for the NTT DOCOMO winter/spring 2011-2012 models (**Photo 1**). This function simplifies access to network services and built-in terminal functions from the standby screen, which is a major point of contact between the user and mobile terminal.

For these new models, we also developed a function for enhancing NTT DOCOMO's Osaifu-Keitai (mobile e-cash) service. This function enables the FeliCa function to be directly accessed from the i-mode browser to shorten the path to services and improve operability while significantly reducing development and operating costs for service providers.

Additionally, as the amount of data that can be stored within current i-appli functions is somewhat small making it difficult to create rich, data-intensive content, we have made i-appli function extensions in these new models to improve the expressive power of i-applis.

In this article, we describe the Palette UI function, in-browser FeliCa access function and i-appli function extensions.

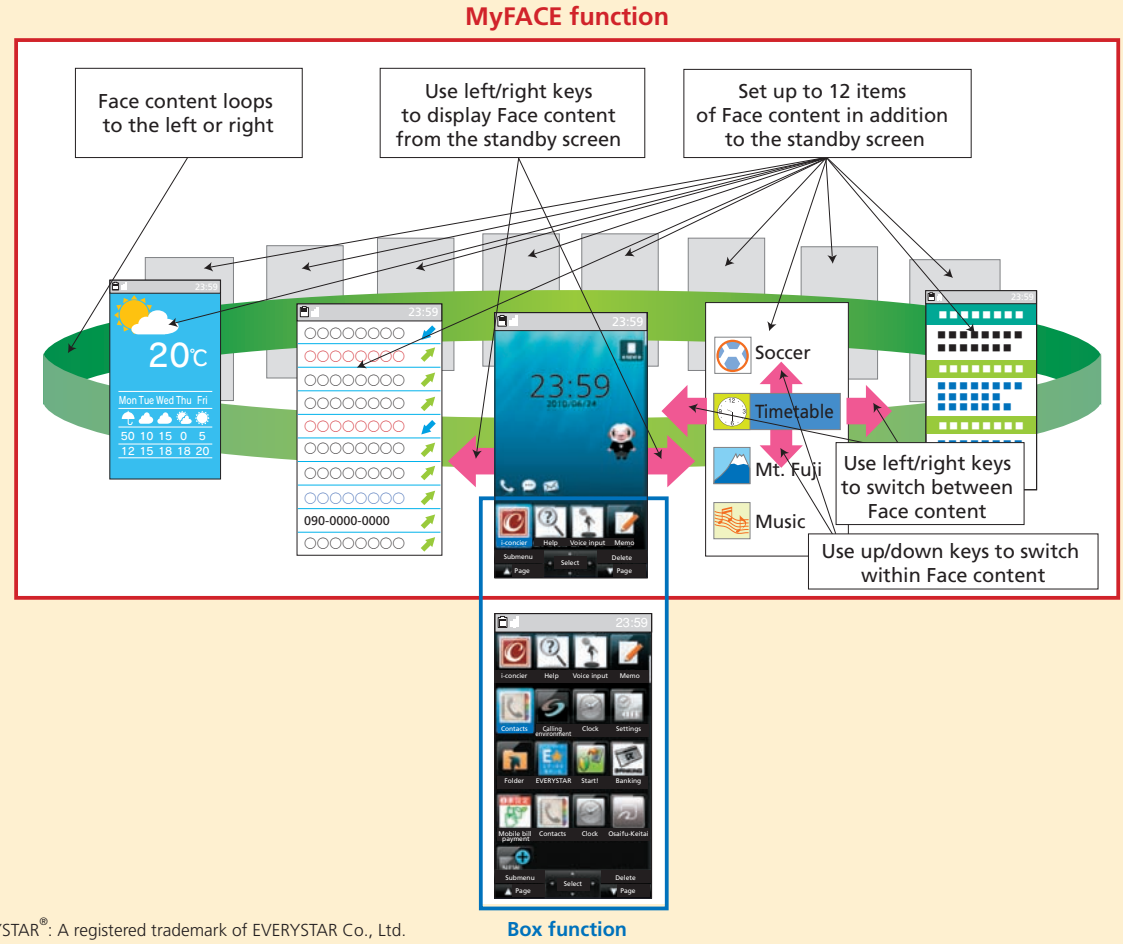
2. Palette UI

2.1 Overview

The basic concept of Palette UI is shown in **Figure 1**. Palette UI consists of a function for virtually pasting a sequence of Web content to the left or right of the standby screen (hereinafter referred to as "MyFACE function") and

*1 **FeliCa®**: A contactless IC card technology developed by Sony Corporation and a registered trademark of the same.

Photo 1 Winter/spring 2011-2012 models



*1 EVERYSTAR®: A registered trademark of EVERYSTAR Co., Ltd.

Figure 1 Overview of Palette UI

a shortcut function that operates in the downward direction from the standby screen and features a means of interaction common to different models (hereinafter referred to as "Box function"). These functions improve accessibility to Web sites, on-terminal services (i-appli, i-concier, etc.), and a variety of functions (contact list, bookmarks, scheduler, alarm, etc.) as starting points on the standby screen.

2.2 MyFACE Function

The MyFACE function is an aggregator of Web content (hereinafter referred to as "Face content") that can be viewed from the standby screen in sequence by using the left/right keys on the terminal or performing left/right flicks on the touch panel. The user can view automatically updated information much like changing channels on a TV and can smoothly connect to Web services or terminal functions from each item of Face content.

The process flow for adding Face content is shown in **Figure 2**. The user can download and add desired Face content from the i-mode browser.

Users can sort and delete downloadable Face content as desired, and can have Face content automatically updated with new information by enabling this function through an appropriate setting. An icon indicating that Face content has been updated with new information will be displayed on the standby screen so that the user can make use of up-to-date information as soon as it becomes available.

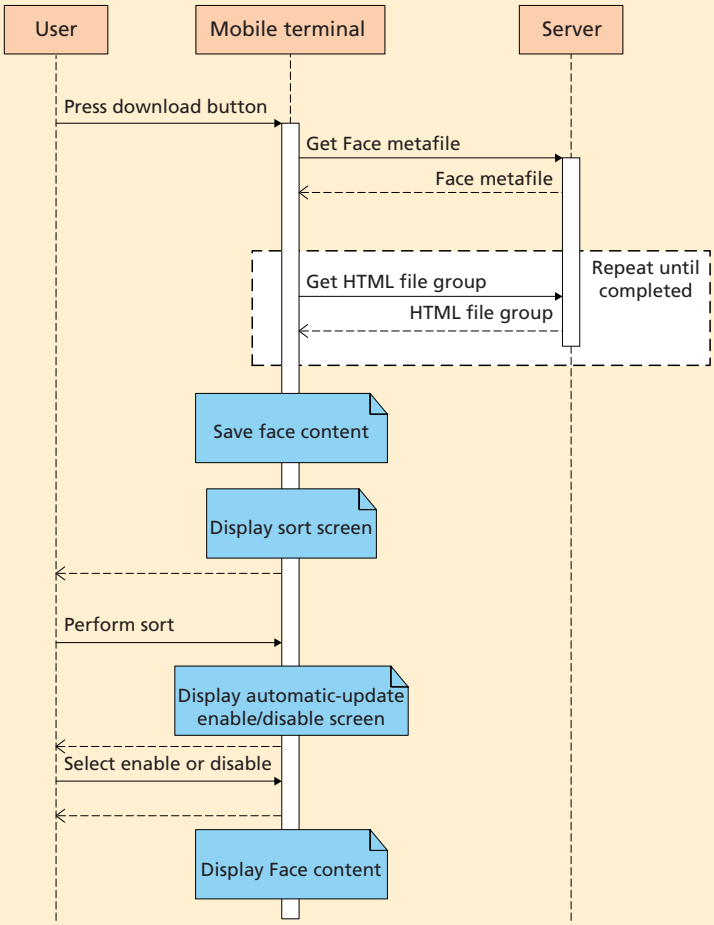


Figure 2 Process flow for adding Face content

Face content can be moved between mobile terminals via infrared communications, iC communications[*2], or microSD™[*3] memory card, it can include linked launching to i-mode browser, full browser, One Seg, telephone, mail, or i-appli applications, and it can be easily introduced to family members or friends by referencing the URL to Face content in mail linked launching.

The configuration of an item of Face content is shown in **Figure 3**. Face content consists of a Face metafile holding Face-content settings (eXtensible Markup Language (XML)[*4] file) and an HTML file group (Web content consisting of HTML/XHTML, JavaScript[*5], Cascading Style Sheets (CSS)[*6], text, images and Flash®[*7]). The use of an HTML file group in this way makes it possible to appropriate existing i-mode browser functions and to therefore reduce the cost of mobile-terminal development and remove barriers to participation by content providers.

*2 **iC communications**: Data communication functions that can be used by simply holding up Felica-equipped mobile terminals next to each other.
*3 **microSD™**: A trademark of SD Card Association.
*4 **XML**: A markup language for indicating meaning and structure by enclosing documents and data in character strings called tags. HTML is another well-known markup language used for configuring Web sites, but its tags are predefined while XML allows the user to specify original tags.
*5 **JavaScript**: A script language appropriate for use in Web browsers.

1) Face Content Rendering

The methods used for rendering Face content are shown in **Figure 4**. Specifically, the winter/spring 2011-2012 models implement Face functions by using the i-mode-browser rendering engine for rendering Face content and the i-concier and scheduler rendering engine for switching Face content and displaying screens for sorting.

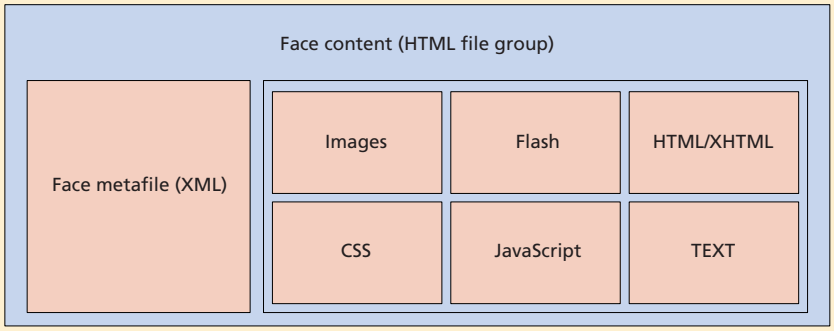


Figure 3 Configuration of Face content

Face content consists of the HTML file group as described earlier, and the specifications of those files support the specifications of i-mode browser 2.1. This means that the functions of this browser—such as external-application linking and Javascript (Ajax[*8]), CSS, Cookie[*9] and Flash—can be used as-is enabling rich expression of content. It also means that Face content inherits the high security level of i-mode browser 2.1 so that threats to mobile phones—which are starting to

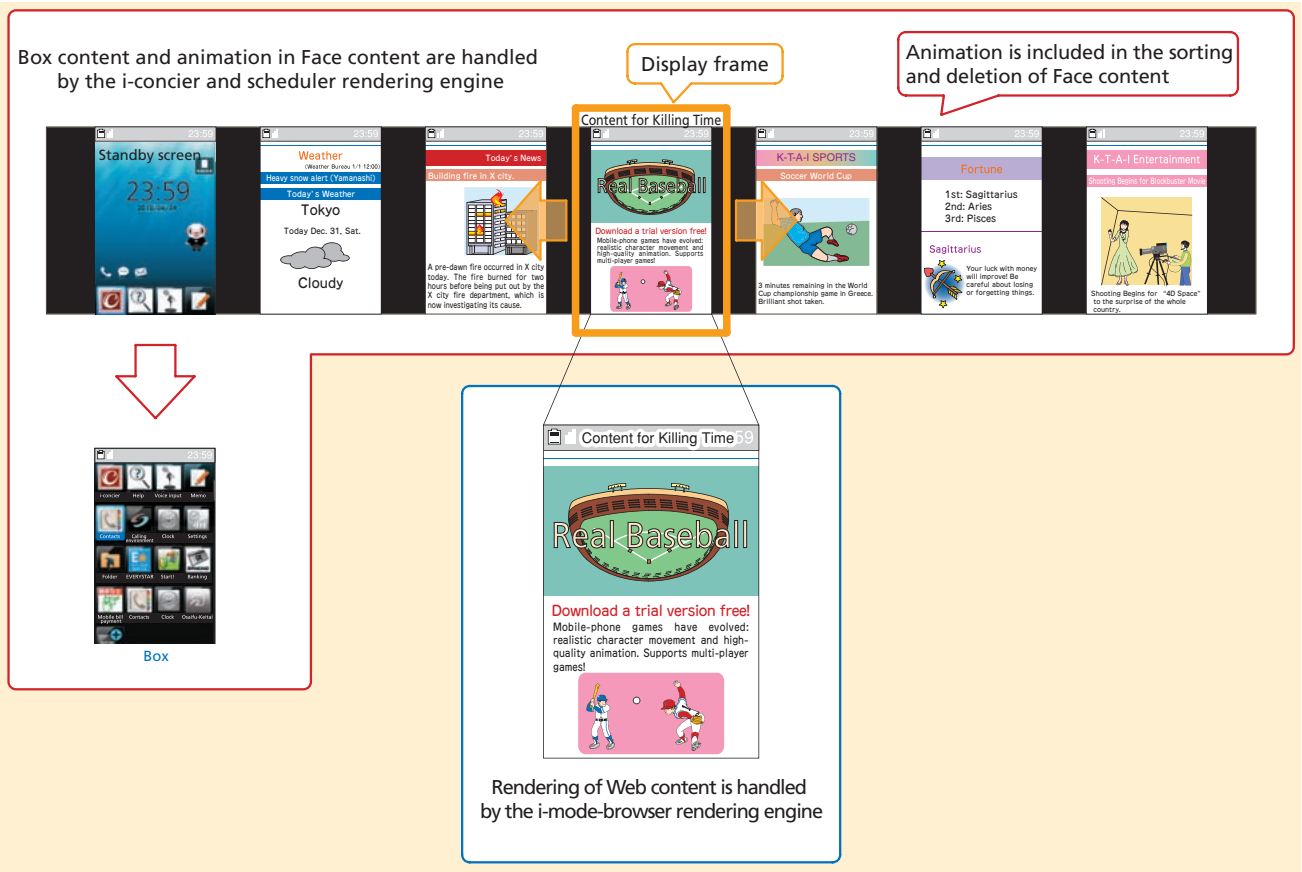


Figure 4 Implementation scheme for rendering of Face content

*6 **CSS**: Specifications used in HTML, XML, and other markup languages for describing how various elements are to be expressed (displayed).

*7 **Flash®**: A type of content developed by Macromedia (now Adobe Systems) that combines audio and animation based on vector graphics (file extension: .swf). Flash® is a trademark or registered trademark of Adobe Systems Inc. in the United States and other countries.

*8 **Ajax**: A format for implementing an interactive Web application that facilitates processing by using the HTTP communication functions of JavaScript in the Web browser and exchanging XML-formatted data with the server without having to reload the Web page.

become an issue—can be prevented and users can enjoy safe and secure mobile communications.

In the development of the MyFACE function, one of the most important criteria for determining whether it could be achieved was the level of response to user actions in making a transition from the standby screen to Face content or from one item of Face content to another. With this in mind, we decided to omit the HTML-file rendering process in the transition from one screen of Face content to another by using a captured image to display Face content. This approach speeds up the screen-transition process.

2) Automatic Updating

We developed an automatic updating function to provide the user with up-to-date information. The concept of temporal dispersion in automatic updating is shown in **Figure 5**. To update content, the terminal refetches content during each of the update hours (0:00 – 23:00) specified in the user's Face metafile (multiple update hours may be specified). However, if mass updating were to occur across the network at exactly the same time, the load on the network would rise, and to prevent this, automatic updating for any one terminal is made to occur at a specific time in minutes and seconds (00:00 – 59:59) within each of the specified hours. This specific time is randomly determined by either the time of terminal manufacture or terminal initialization. This disperses update processing thereby reducing the load on the network.

3) Security

The scheme implemented for pro-

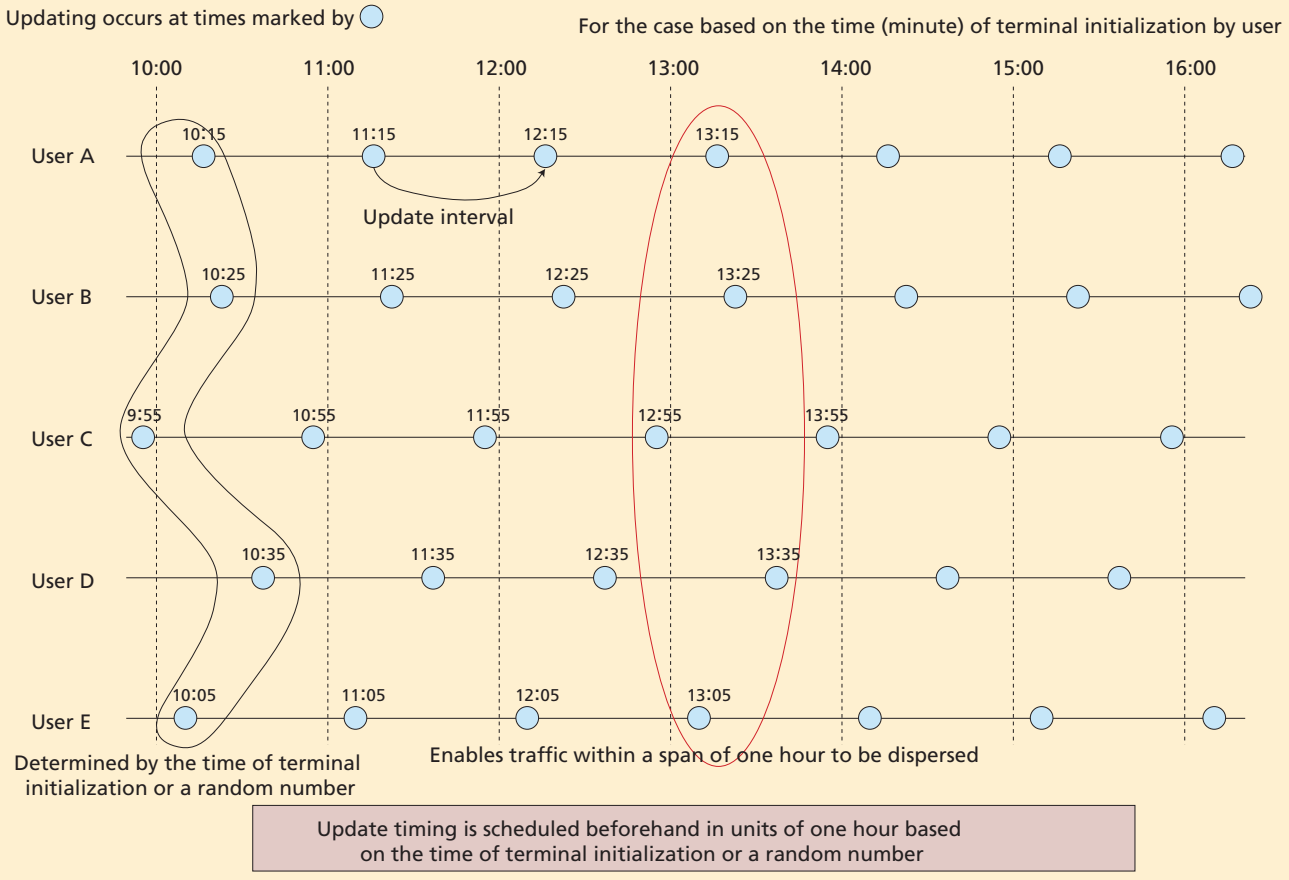


Figure 5 Temporal dispersion of automatic updating

*9 **Cookie**: A function for storing user information, the lengths and number of visits to a site, and other data on the terminal to make interaction with that site more convenient for the user.

tecting the rights of Face content creators is shown in **Figure 6**. In this scheme, the URL of the HTML file group is described in the Face metafile. This creates concern that an author with malicious intent could use the URL of another person's HTML file group without permission and distribute that Face content as his own. To prevent this from happening, we have implemented a function that first compares the URL for retrieving the Face metafile with the URL of the HTML file group when a user attempts to download Face content and then enables the HTML file group to be fetched only when those URLs are in the same directory of the same domain (or when one URL is in a subdirectory of the other). This scheme protects the rights of the Face content creator.

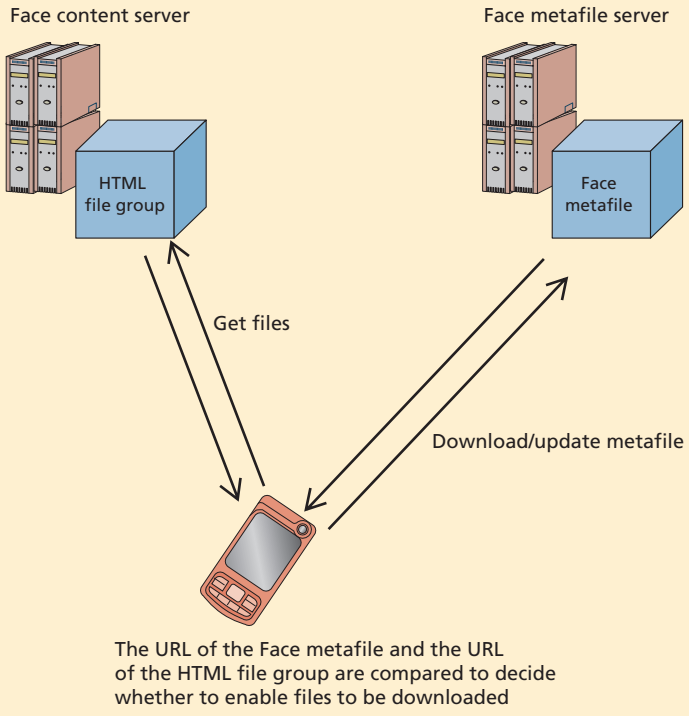


Figure 6 Scheme for protecting the rights of Face content creators

2.3 Box Function

The "Box" is a function that enables shortcuts to frequently used services and functions to be arranged as desired on a virtual screen extending from the bottom of the standby screen. It can also display shortcuts for the four most frequently used services and functions at the bottom of the standby screen.

The Box function also incorporates a separator line having a folder function to simplify folder management of shortcuts. This separator line features an accordion display function that enables a folder to be open or closed with one action without changing screens, which makes it easy to hide shortcuts that are currently unneeded.

Furthermore, by automatically displaying a new separator line at the bottom of the Box when the user attempts to move a shortcut to another folder, there is no need for the user to perform a separator-line add operation thereby easing the burden on the user.

3. In-browser FeliCa Access Function

3.1 Background to Development

In NTT DOCOMO's Osaifu-Keitai service, the FeliCa chip is accessed by applications (hereinafter referred to as "IC-applis") that have traditionally been developed individually by service providers. This has meant that users who wished to use Osaifu-Keitai functions have had to download an IC-appli for each IC service from the Web page of that service provider and then launch and initialize it, which was hardly a convenient way to start using a new service. Many users have cried out for a less complicated procedure for getting started with an Osaifu-Keitai IC service and developing such a procedure has come to be seen as a necessity for the sake of expanding services.

From the viewpoint of service providers, however, providing IC-applis that achieve secure access to FeliCa chips and prevent data loss or leaks is a

necessity since these chips store data equivalent to cash. As a consequence, the development and maintenance of IC-applis has been a major barrier to providing services for both existing and new service providers.

To solve these two issues, we developed a new function for accessing the FeliCa chip from the Web page of each service provider via a preinstalled IC-appli (hereinafter referred to as "common IC-appli") that can be used in common by different IC services.

This function makes it unnecessary for each service provider to develop and maintain an IC-appli thereby freeing up the provider to devote its efforts to developing and enhancing its Web page. As a result, barriers to participating in and operating an Osaifu-Keitai service can be greatly reduced. Users also benefit by not having to download an IC-appli for each service beforehand, which greatly improves the convenience of using Osaifu-Keitai.

3.2 Presentation of IC Services

Introducing this common IC-appli, however, raises the question of how to present to the user the screens for checking account information and transaction history and charging for a service. Since the conventional approach was to have each IC-appli issue one corresponding IC service, each IC service was displayed in units of IC-applis on the IC Card List screen on the terminal. With the new function, however, the common IC-appli issues multiple IC services, which prevents IC services from being displayed individually to the user by the conventional display method.

In response to this issue, we developed a new function for reading the content of the FeliCa chip and for identifying and displaying the IC services that are actually being issued. The basic concept of this IC-service display function is shown in **Figure 7**. As shown in the figure, selecting the Online Service List at the top of the IC Card List screen causes the system to switch to the Online Service List screen. The IC

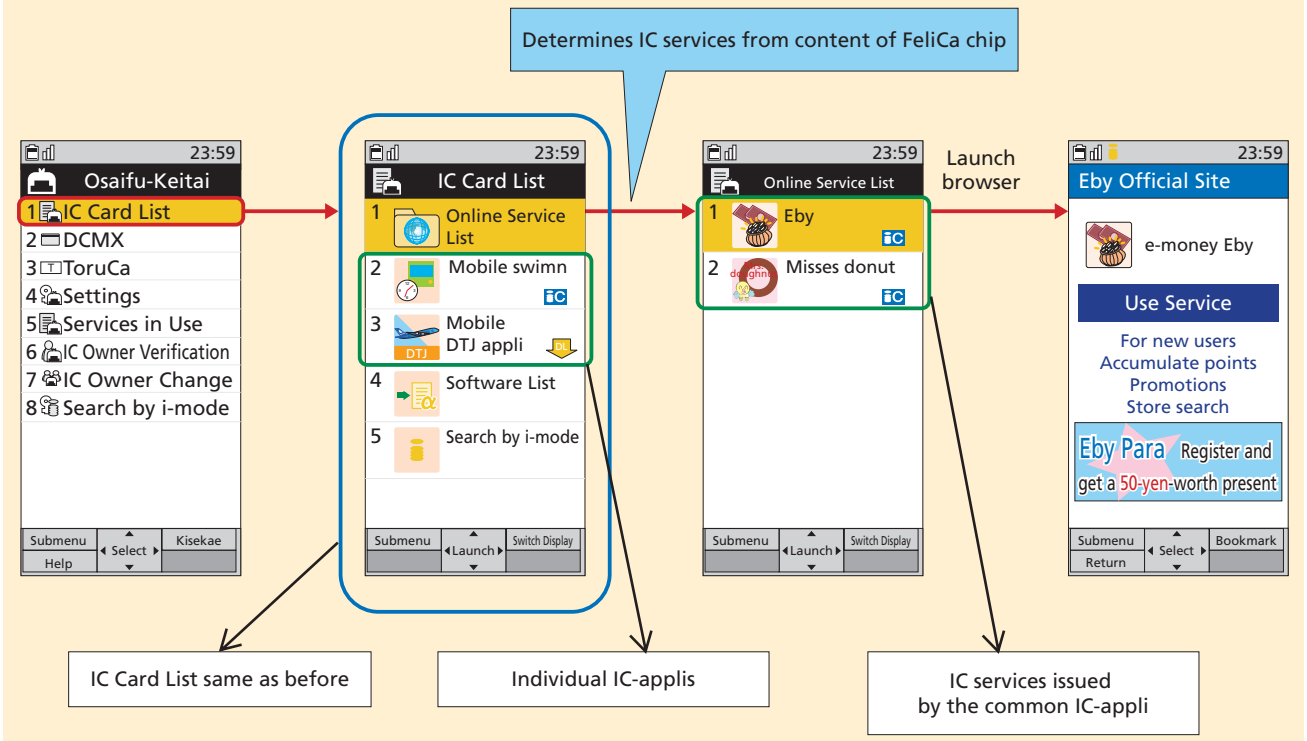


Figure 7 Concept of IC service display

services displayed on this screen are the ones issued by the common IC-appli.

This function makes it possible to provide a presentation format on the same level as that of conventional IC services.

3.3 IC Service Moving Function for Non-supporting Terminals

Another important function in Osaifu-Keitai is the iC Ohikkoshi Service for moving IC services from a user's current terminal to a new one. However, there are older terminals that do not support the FeliCa-chip access function using the common IC-appli, which means that moving IC services between a non-supporting terminal and a supporting terminal requires that the differences between the two terminals be absorbed when relaying the information. Specifically, moving IC services from a non-supporting terminal to a supporting terminal requires that they be moved from the individual IC-applis to the common IC-appli, and moving IC services from a supporting terminal to a non-supporting terminal requires that they be moved from the common IC-appli to the individual IC-applis.

With this in mind, we developed a function within ALl Around DOCOMO InformatioN systems (ALADIN)[*10] to support such movement of IC services, which has the effect of maintaining compatibility between non-supporting and supporting terminals.

In short, this function makes it possible to move IC services between a terminal that does not support the in-browser FeliCa access function and a terminal that does. As such, it is expected to motivate both users and service providers to migrate to terminals that support the in-browser FeliCa access function.

4. i-appli Function Enhancements

4.1 Background to Development

Creating rich applications using large amounts of data was difficult on the past i-appli platform [2]. There were two main reasons for this. First, capacity limitations of the i-appli itself made it difficult to store data-intensive resources within the i-appli, and second, the use of data storage outside the i-appli was also limited. In regard to the first point, the maximum size of the Java Archive (JAR)[*11] file and scratchpad[*12] combined for an i-appli was 2 MB, which made it difficult to store resources such as large-capacity audio and video files within the i-appli. In regard to the second point, there have been four methods for storing resource files used by an i-appli: data BOX[*13], scratchpad, server and external memory (microSD card, etc.). Each of these methods, however, has its own limitations or constraints. The data BOX, for example, can only read or write specific types of files, and its use is hampered by the display of data-selection and user-verification screens requiring user actions. Using a server for storing resources, meanwhile, provides essentially limitless capacity, but since resources can only be retrieved via wireless communications, the user must pay packet communication fees and must be in a location with good signal conditions to access those resources. There are also practically no capacity limits in the case of external memory, but data read/write time is longer compared to internal memory and the user must acquire and attach an external memory device.

Both users and content providers have been expressing for some time a desire for improvements to overcome these limitations.

We therefore implemented the function enhancements described below in the winter/spring 2011-2012 models to enable the creation of rich, data-intensive i-applis. These enhancements have made it possible, for example, to provide a mail-appli that stores mail data within the mobile terminal and a comic-appli that stores comic data in the terminal's internal storage. We have also updated the version of the Star profile from Star-1.5 to Star-2.0 so that these i-appli function enhancements can be far reaching in their effect.

4.2 i-appli Capacity Expansion

For the new models, we expanded the size of the i-appli JAR file and scratchpad combined from the current 2 MB to 10 MB. Expanding the capaci-

*10 **ALADIN**: A customer management system.
*11 **JAR**: A file type in which Java byte codes that have been generated by compiling Java source code files are combined into one archived file.
*12 **Scratchpad**: A data storage area secured in a memory device within the mobile terminal for use by an i-appli. Scratchpads are allocated to individual i-applis, and a scratchpad allocated to one i-appli cannot be accessed by another i-appli. Data continues to be stored after the i-appli terminates execution. The user cannot view scratchpad data.
*13 **Data BOX**: A storage area for images, video and other types of data in FOMA terminals. The user can view data inside the data BOX.

ty of the i-appli, however, generates new issues as it lengthens the start-up time of the i-appli and increases the amount of heap memory[*14] used. We addressed the first issue by displaying an appli-specific start-up image as described below to reduce the user's sense of waiting for the i-appli to launch, and we addressed the second issue by expanding the amount of heap memory.

4.3 Saving of i-appli Data in Internal Storage

To avoid the limitations described above when using external data storage areas, we made it possible to save resource files used by the i-appli in the mobile terminal's internal storage. A typical layout of data storage areas is shown in **Figure 8** and the functions of these data storage areas are compared in **Table 1**. Internal storage capacity, while depending on the size of the terminal's internal memory, is generally suitable for handling several tens of MB. An example of how folders and files might be hierarchically arranged in internal storage is shown in **Figure 9**. The 1st layer corresponds to the root folder[*15] and the 2nd layer to exclusive folders for each i-appli. In this scheme, each i-appli stores data in its own exclusive folder and cannot read or

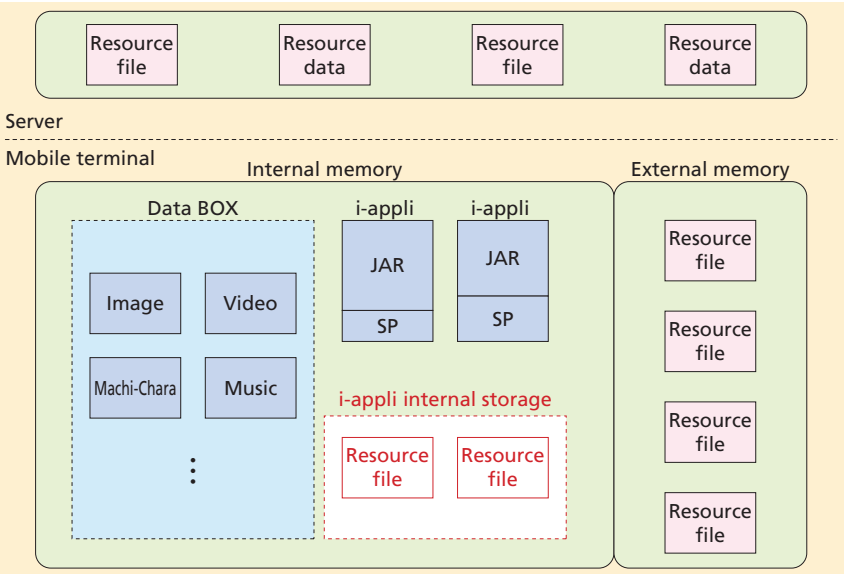


Figure 8 Typical layout of data storage areas

Table 1 Function comparison of data storage areas

	Data Storage Areas				
	Data BOX	Scratchpad	Internal Storage	Server	External memory
Capacity	△ · Depends on upper limit of mobile terminal's internal memory	× · 10 MB with JAR combined	△ · Depends on upper limit of mobile terminal's internal memory	◎ · Depends on upper limit of server capacity	○ · Depends on external-memory capacity
Read/write speed	× · Involves time-consuming user actions	◎	◎	△ · Depends on signal conditions	○ · Depends on external-memory standard
Ease of use	× · Can read or write only specific types of files · Includes display of data-selection and user-verification screens requiring user	◎	◎	× · Generates user packet communication fees · Can use only from a location with good signal conditions	× · Requires user to acquire and attach external memory

◎ very good ○ good △ not good × bad

*14 **Heap memory**: A type of memory area used by OS and application software.

*15 **Root folder**: The uppermost folder in a folder hierarchy.

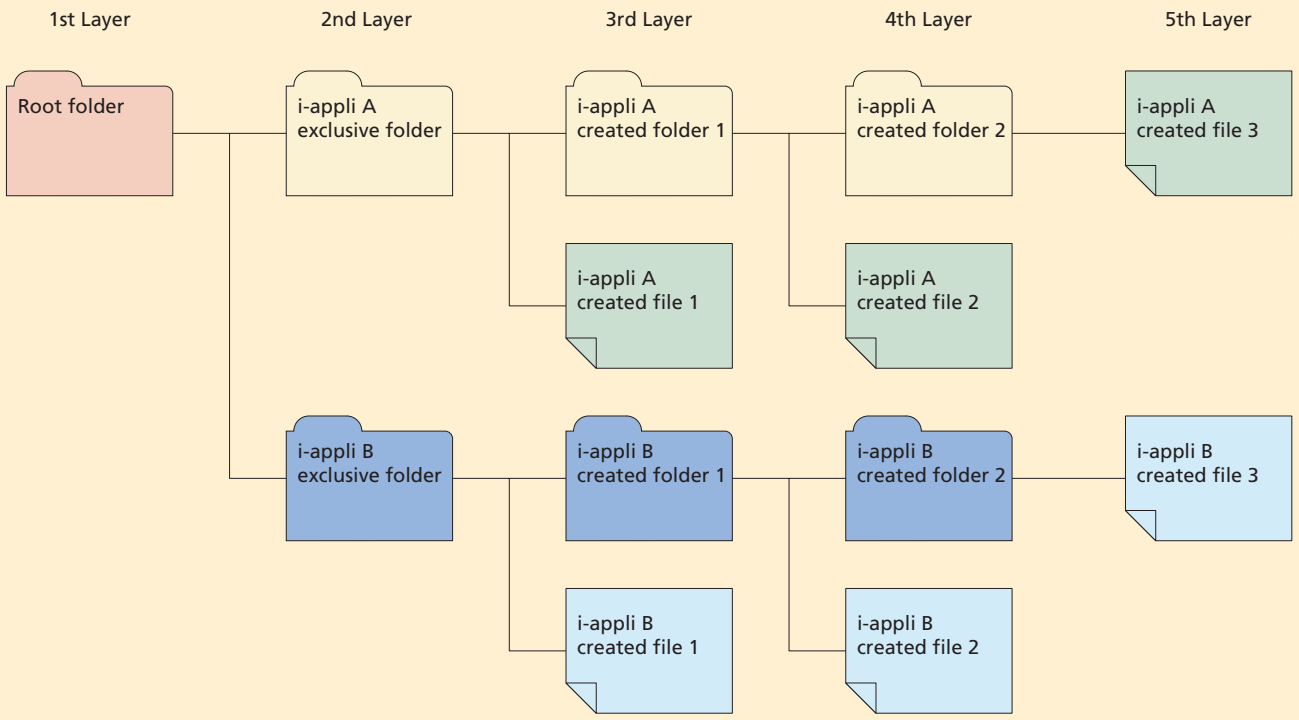


Figure 9 Typical arrangement of folders and files on each layer of internal storage

write data from or to another appli's exclusive folder. Data is saved in the form of a file with no restrictions placed on the file extension. Any i-appli data stored in internal storage will not appear in the mobile terminal's data BOX.

4.4 PNG Support

Image types that can be used by i-applis have in the past been limited to JPEG, Graphic Interchange Format (GIF)[*16], and Bitmap (BMP)[*17]. If transparency processing was desired, it was necessary to use GIF. The issue here, however, is that the number of colors that GIF can handle is relatively small, which has prevented transparency processing from being applied to high-resolution images in i-applis.

To address this issue, we enabled i-applis in the winter/spring 2011-2012 models to use Portable Network Graphics (PNG)[*18], which is capable of high-resolution transparency processing. This implementation supports transparency specifications based on the alpha[*19] and tRNS chunk[*20] parameters.

In addition to i-applis, the PNG image format can also be used for icon and thumbnail images in software lists and main-title images.

4.5 Font Improvements

In the past, only seven standard font sizes (two-byte character 12×12 dots, 16×16 dots, 24×24 dots, 30×30 dots, 32×32 dots, 48×48 dots and 60×60 dots) and those uniquely supported by terminal manufacture could be specified by i-applis, which has meant design constraints for i-appli developers.

To rectify this issue, we increased the font-size range in the new models. Developers can now specify fonts in the range from two-byte character 12×12 dots (one-byte character 6×12 dots) to two-byte character 60×60 dots (one-byte character 30×60 dots) in 1-dot units.

4.6 Start-up Images

When starting up an i-appli, the past

*16 **GIF**: An image file format that can represent up to 256 colors.
*17 **BMP**: An image file format that can represent up to approximately 16,780,000 colors. It is an uncompressed format, which means that BMP files tend to be larger than those of other formats.
*18 **PNG**: An image file format that can represent up to approximately 280 trillion colors with transparency processing.
*19 **Alpha**: A parameter associated with transparency processing and stored in a PNG file.
*20 **tRNS chunk**: A parameter associated with transparency processing and stored in a PNG file.

method was to display a common image regardless of the i-appli, but displaying an image unrelated to the i-appli had the unfortunate effect of diminishing the concept of i-applis.

For these new models, we have made it possible for an i-appli's start-up image to be set from the i-appli so as to express as much as possible the concept of i-applis. Here, two types of start-up images can be set according to screen orientation: one for portrait view and the other for landscape view. The file format of the start-up image may be JPEG, BMP, GIF (including animation GIF), or PNG. The vertical/horizontal size of the image is unrestricted and the image is displayed in the center of the screen without expansion or contraction.

5. Conclusion

This article described the Palette UI function, in-browser FeliCa access function, and i-appli function extensions for the NTT DOCOMO winter/spring 2011-2012 models. These new applications shorten the user's access path to terminal functions and network services and achieve a more convenient, easier-to-use platform. Looking to the future, we plan to continue our application-development efforts as services and user needs become even more diversified.

REFERENCES

[1] NTT DOCOMO: "Let's Make i-mode: i-mode Browser 2.1—Service and Functions—NTT DOCOMO". http://www.nttdocomo.co.jp/english/service/developer/make/content/browser/browser2_1/index.html

[2] E. Oseki et al.: "Application Functions for Autumn/Winter 2008 Models (2) Next-generation Java Applications," NTT DOCOMO Technical Journal, Vol. 10, No. 4, pp. 51-59, Mar. 2009.