

[print](#) | [close](#)

APIs by Example: User Application Information APIs

[System iNetwork Programming Tips Newsletter](#)

[Carsten Flensburg](#)

Carsten Flensburg

Thu, 03/23/2006 (All day)

Have you ever wondered where the system stores information about a command's previous assistance level, or Query/400's last-used query and file libraries, or PDM's default options? Have you ever needed to store your own settings on a per-user basis in one of your applications? If the answer is yes to any of these questions, please read on, because this and a future installment of APIs by Example provide information about these topics.

Let's begin with storing application-specific data for persistent options and settings for a specific user profile. In V5R3, IBM added three User Application Information APIs that let you store, retrieve, and remove this type of information in a consistent and robust way, throughout all your applications.

The three APIs are:

- Update User Application Information, which creates or updates the specified User Application Information ID and related data for a given user profile.
- Retrieve User Application Information, which retrieves one or more User Application Information IDs and related data for a given user profile.
- Remove User Application Information, which removes one or more User Application IDs for a given user profile.

As an added bonus, all the application data stored using the User Application Information APIs is saved and restored together with the user profile that it belongs to. Because of this, when you update user application information, you have to specify the earliest release in which the information is valid. That way, the system knows not to restore it to a release in which it's invalid.

It is also important to understand that to update or remove user application information for another user profile, you must have *SECADM special authority and *OBJMGT and *USE authorities to that user profile. To retrieve the information, *READ authority to the user profile is required. Follow the link at the end of this article to read all the details about the User Application Information APIs.

Today, I present a service program that offers a simplified interface to the APIs as well as helps control and centralize the setting of some of the API parameters. In the next installment, I will create a sample application, including three commands, that uses the User Application Information APIs to control a user's time zone setting and thereby the display of date and time information for that user.

The service program exports the following set of User Application Information API interfacing procedures, with the aim of simplifying the use and parameter structure of these APIs:

```
SetUsrAppInf() -- Creates or updates user application information
VfyUsrAppInf() -- Verifies user application information
```

```
GetUstrAppInf() -- Retrieves user application information
DltUstrAppInf() -- Removes user application information
```

Further, another set of time zone–specific user application information procedures are exported. They are based on the preceding procedures, but their parameter structure is directly targeted at storing and retrieving time zone information for a given user profile. A time zone information data structure documenting the layout of this information is also included. As I mentioned, these procedures provide the basis of the sample application in the next installment of this article:

```
SetUstrTimZon() -- Creates or updates user time zone information
VfyUstrTimZon() -- Verifies user time zone information
GetUstrTimZon() -- Retrieves user time zone information
DltUstrTimZon() -- Removes user time zone information
```

To allow for immediate verification of the service program, I wrote a short sample program to perform a brief test of the service program's exported general user information procedures. To avoid authorization issues, the test is performed using the current user profile of the job. It performs the following steps:

1. Display the name of the job's current user profile.
2. Prompt you to enter a text string to store with the user profile.
3. Verify the successful update of the user application information.
4. Retrieve and display the stored user application information.
5. Remove the user application information.
6. Verify the successful removal of the user application information.

By then we should be ready to continue our exploration of the User Application Information APIs in the next installment of APIs by Example.

And now on to the other question, regarding the location of user profile assistance level, query libraries, PDM options, and many other profile information entries: They are all stored as part of the user profile in a so-called independent index. The following command, which requires *ALLOBJ special authority, dumps the user profile's internal profile information index to a spooled file:

```
DmpSysObj  Obj ( )
           Context( QSYS )
           Type( 0E )
           SubType( C4 )
```

No public documentation exists for the layout of the index entries, so to find out precisely what information is stored where, some trial-and-error efforts are required. But I'm sure that the dump will give you an immediate idea about the type and format of information stored.

You can also retrieve and manipulate this information programmatically, using MI-independent index built-ins, but because the index is off-limits to user-state programs, this can be done only at security level 30 or below. And please note that at that security level, security audit journal entries are generated to register the domain violation, if the audit journal is active on the system running these instructions. To check, look for journal code "T" and entry type "AF."

For anyone who has this opportunity, I include a small program that retrieves a user profile's PDM option settings. Look in the source header of the CBX152MI program for further instructions.

This APIs by Example includes the following sources:

```
CBX152    -- User application information service program
CBX152B   -- Service program binder source
CBX152T   -- Sample program verifying service program
CBX152MI  -- Sample program retrieving PDM options
```

As usual, compilation instructions are in the source headers.

This article demonstrates the following APIs:

User Application Information APIs:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/misc1c.htm>

Update User Application Information (QsyUpdateUserApplicationInfo) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/qsyupdui.htm>

Retrieve User Application Information (QsyRetrieveUserApplicationInfo) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/qsyrvtvui.htm>

Remove User Application Information (QsyRemoveUserApplicationInfo) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/qsyrmvui.htm>

Retrieve Product Information (QSZRTVPR) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/qszrtvpr.htm>

Send Program Message (QMHSNDPM) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/QMHSNDPM.htm>

Receive Program Message (QMHRCVPM) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/QMHRCVPM.htm>

Display Long Text (QUILNGTX) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r3/topic/apis/quilngtx.htm>

You can retrieve the source code for this API example from

http://www.pentontech.com/IBMContent/Documents/article/52288_62_UserAppInfo1.zip.

Source URL: <http://iprodeveloper.com/rpg-programming/apis-example-user-application-information-apis>