


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

APIs by Example: The Save to Application API and Exit Program

[System iNetwork Programming Tips Newsletter](#)

[Carsten Flensburg](#)

Carsten Flensburg

Thu, 05/22/2008 (All day)

At first glance, the Save to Application (QaneSava) API looks complicated, and its purpose might seem unclear. The QaneSava API lets you intercept the save file records generated by one of up to eight different save commands or APIs. Instead of directing the save records to a save file, the QaneSava API intercepts and routes the save records to an API-parameter-defined, user-written exit program. In a moment, I discuss how this is done. In addition, getting direct and immediate access to the save file records is useful for at least a couple of reasons, which I outline in this article.

Using the QaneSava API, you could transfer the save records to another system or site as the save process unfolds, instead of having to wait until the save process has completed. Another option would be to encrypt the save records before storing them on a medium of your choice. The QaneSava API of course has a counterpart in the QaneRsta (Restore from Application) API, which lets you reverse the process, bringing the entire save/restore process within reach of your programming skills. Today, I focus on the save perspective, and to demonstrate the QaneSava API and the related Save to Application Exit Program, I created the Save Object to Stream File (SAVOBJSTMF) command.

As I just mentioned, other options for transferring or encrypting a save file already exist. One of the more straightforward options is the Copy (CPY) command. After a save file is created and the save process is completed, you could, for example, use the following command to copy the save file SAVE001 in library QGPL to the QOpenSys file system in the IFS on your machine:

```
CPY      OBJ( '/QSYS.LIB/QGPL.LIB/SAVE001.FILE' )
          TODIR( '/QOpenSys' )
          DTAFMT( *BINARY )
```

This command would produce a stream file in the QOpenSys directory:

```

Work with Object Links

Directory . . . . : /qopensys

Type options, press Enter.

2=Edit   3=Copy   4=Remove   5=Display   7=Rename   8=Display
```

```

attributes
  ll=Change current directory ...

Opt   Object link           Type   Attribute   Text
      .                  DIR
      ..                 DIR
      SAVE001.FILE        STMF    SAVF

```

From this point on, you can transfer and process the stream file as you require, as long as you can reverse the process when you need to copy the stream file back to its save file. To achieve the final step, run a CPY command like this one:

```

CPY      OBJ( '/QOpenSys/SAVE001.FILE' )
         TOOBJ( '/QSYS.LIB/QGPL.LIB/TEST001.FILE' )
         DTAFMT( *BINARY )

```

If you'd rather pursue the API and exit program route, however, you'll be interested to know that the details involved in choosing that approach are herein. The QaneSava API has the following parameter list:

1	Qualified user space name	Input	Char(20)
2	User space format name	Input	Char(8)
3	Status format name	Input	Char(8)
4	Status information	Output	Char(*)
5	Length of status information	Input	Binary(4)
6	Error code	I/O	Char(*)

The first parameter specifies the name and library of a user space that must contain all the control information for the save operation. This information is formatted as a data structure named SVRS0100 and includes such information as the save command or API to be used to generate the save records, the save command parameters or save API parameter key structure, and a section called *Application data*, available to support communication between the program calling the QaneSava API and the exit program called during the save process.

The name and library of the exit program to be called is also defined by the SVRS0100 data structure. Here's the manual's description of the SVRS0100 data structure in its entirety:

Offset	Type	Field
Dec	Hex	
0	0	BINARY(4) Length of structure
4	4	BINARY(4) Offset to save command parameters
8	8	BINARY(4) Length of save command parameters
12	C	BINARY(4) Offset to application data
16	10	BINARY(4) Length of application data
20	14	BINARY(4) Save command type

24	18	CHAR(10)	Exit program name
34	22	CHAR(10)	Exit program library
44	2C	CHAR(8)	Target release
		CHAR(*)	Save command parameters
		CHAR(*)	Application data

The save commands and APIs supported by the QaneSava API are defined by the following list, which shows all available options at release V5R4 for the Save command type subfield in the preceding structure:

1. Save (SAV) command
2. Save Object (SAVOBJ) command
3. Save Document Library Object (SAVDLO) command
4. Save Library (SAVLIB) command
5. Save Changed Object (SAVCHGOBJ) command
6. Save Object (QsrSave) API
7. Save Object List (QSRSAVO) API
8. Save System Information (SAVSYSINF) command

Note that the SVRS0100 parameter structure also contains the name and library of the exit program to be called during the save process. The name of the parameter structure is submitted as the second API parameter.

The QanaSave API's third, fourth, and fifth parameters define the name of the status data structure, the status data structure itself, and the length of the status data structure, respectively. The status data structure is named SRST0100 and provides a communication area between the QaneSave API and its caller and contains the following information at release V5R4:

Offset	Type	Field	
Dec	Hex		
0	0	BINARY(4)	Bytes returned
4	4	BINARY(4)	Bytes available
8	8	BINARY(4)	Transfer time
12	C	BINARY(4)	Transfer block size
16	10	BINARY(4)	Transfer block multiplier
20	14	BINARY(4)	Last block size
24	18	CHAR(10)	User space library used
34	22	CHAR(2)	Reserved
36	24	BINARY(4)	Decimal transfer time

The final and sixth parameter is the standard API error code data structure, which I assume that you are already familiar with. If you're not, I urge you to read Scott Klement's article titled ["A Beginner's Guide to APIs"](#). This article offers a great introduction not only to the API error code data structure but to APIs in general.

The QaneSava API has a lot of restrictions, constraints, and subtleties to consider when you code this API. For example, if the save command is submitted by a prestart job and not in the job that called the API, some save command parameters are not supported for various reasons, and objects saved by the QaneSave API can only be restored using the QaneRsta API. All these issues are thoroughly described in the API documentation, so I'll quit copying the documentation for now and refer you to the link below, following which you'll find the IBM online V5R4 version of the QaneSava API documentation.

Be sure to also check out the code included with this APIs by Example to see how the pieces fit together, and feel free to send me any questions that remain following your reading exercise.

So back to the second component in today's utility: The Save to Application Exit Program. As I noted earlier, you register your exit program in the SVRS0100 data structure, and following a successful call to the QaneSava program, this exit program gets called in any of the following four operation types:

1. Start of save process. At this point, the exit program must prepare for the save record transfer, as for example open a stream file.
2. Transfer of a block of save records. The exit program receives a block of save records as well as the length of the data block. The exit program can then store the received data as for example write or append the data to a stream file. This step repeats until all save records are processed.
3. End of save process. The exit program at this point must terminate the process of storing the save records, and for example, close the stream file.
4. Abnormal end of save process. If anything goes wrong in the cause of the save process, the exit program is called one final time to allow it to perform cleanup and termination activities, such as deleting the save records stream file.

The exit program has the following required parameter group:

1	Operation type	Input	Binary(4)
2	Operation status	Output	Binary(4)
3	Save data	Input	PTR(SPP)
4	Length of save data	Input	Binary(4)
5	Save bytes read	Output	Binary(4)
6	Qualified user space name	Input	Char(20)
7	User space format name	Input	Char(8)

The *Operation type* parameter specifies which of the preceding four events caused the exit program to call, and the exit program returns the outcome of the processing performed by the exit program in the *Operation status* parameter, which in turn causes the save process to either continue or terminate.

The third parameter is a space pointer to a block of save records. This parameter is of course passed only for operation type 2. Together with the *Length of save data* parameter, these two parameters enable you to process the save data buffer safely. The sixth parameter is the qualified name of the user space specified as input to the QaneSava API, and it thereby enables the exit program to share and exchange information with the API caller via this user space. The format of the data in the user space is declared by the seventh parameter.

Summing up all the above information leads finally to the SAVOBJSTMF command, which has the following command prompt appearance:

Save Object to Stream File (SAVOBJSTMF)

Type choices, press Enter.

```

Objects . . . . . Name, generic*,
*ALL
      + for more values

Library . . . . . Name

Object types . . . . . *ALL
      + for more values

Stream file . . . . .

Additional Parameters

Target release . . . . . *CURRENT      *CURRENT, *PRV

Update history . . . . . *YES          *YES, *NO

Save active . . . . . *NO              *NO, *LIB,
*SYNCLIB, *SYSDFN
Save active wait time:

Object locks . . . . . 120             0-99999, *NOMAX

Pending record changes . . . . . *LOCKWAIT 0-99999,
*LOCKWAIT...
Other pending changes . . . . . *LOCKWAIT 0-99999, *LOCKWAIT,
*NOMAX
Save active message queue . . . . . *NONE   Name, *NONE

Library . . . . . *LIBL               Name, *LIBL,
*CURLIB
Output . . . . . *NONE                *NONE, *PRINT

```

The *Stream file* parameter specifies an existing path as well as a stream file created as part of the save processing. The stream file will contain all save records after the save is complete. As always, a help text panel group is included with the command to explain all details and parameters. Apart from the Stream file parameter, all other parameters work the same way as you're used to with the Save Object (SAVOBJ) command.

Please note that if you specify the OUTPUT(*PRINT) keyword for the SAVOBJSTMF command, you can locate the produced spooled file using the command:

```
WRKSPLF SELECT(*CURRENT *ALL *ALL *ALL *ALL QPSAVOBJ)
```

Because the save process runs in a separate prestart job, the spooled file is created under the QPRTJOB special job name and is therefore not owned by the job running the SAVOBJSTMF command, eventually making it a bit harder to locate.

In the next installment of APIs by example, I show you how to reverse the process and restore objects using the Restore from Application (QaneRsta) API and the associated exit program.

This APIs by Example includes the following sources:

```
CBX193H  -- PNLGRP  -- Save Object to Stream File - Help
CBX193X  -- CMD      -- Save Object to Stream File
CBX1931  -- RPGLE   -- Save Object to Stream File - CPP
CBX1931V -- RPGLE   -- Save Object to Stream File - VCP
CBX1932  -- RPGLE   -- Save to Application - Exit program

CBX193M  -- CLP      -- Save Object to Stream File - build command
```

To create all these objects, compile and run CBX193M, following the instructions in the source header. As always, compilation instructions are in the respective source headers.

This article demonstrates the following Backup and Recovery API and Exit Program:

Save to Application (QaneSava) API:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r4/topic/apis/QaneSava.htm>

Save to Application Exit Program:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r4/topic/apis/XANESAVA.htm>

Backup and Recovery APIs V5R4:

<http://publib.boulder.ibm.com/infocenter/iserics/v5r4/topic/apis/back1.htm>

You can retrieve the source code for this API example from:

http://www.pentontech.com/IBMContent/Documents/article/56711_616_SavObjStmf.zip.

Source URL: <http://iprodeveloper.com/rpg-programming/apis-example-save-application-api-and-exit-program>