Advanced API

Paul Roy

API

- Agenda
  - Définition
  - Documentation
  - Tips and tools
    - IFS
    - User spaces…
    - Pointers
    - SQL
    - Lists Asynch

API

- Definition
  - Application programming interface
  - It's just a program, a procedure, a function that allows an application program to interact with operating system functions
    - Execute a system command
    - Retrieve a system information
      - Status, value, etc…

API

- Families of API
  - Simple interactions with i5/OS
    - RTV* commands
    - QCMDEXC
    - DSP* commands with OUTPUT(*FILE)
    - DTAQ operations
  - API for system Administration
    - List objects
    - Spooled files

API

- C Library
  - IFS: Open(), read(), write(), close(), readdir(), etc…
  - MATH: abs(), sin(), cos(),..
  - SOCKET: listen(), accept(), gethostbyname(), gsk..
- Security: Krb5… ldap… eim*
- ENCRYPT : QC3ENCDT, QC3DECDT,…
- HA : QcstAddClusterNodeEntry ,..
- DB: QDBRTVFD..
- MI: (all MI instructions) ADDN, BITPERM, CMPNV, CPYBLA …

API

Documentation

- Start from knowledge center/Api Finder
  - https://www.ibm.com/support/knowledgecenter/ssw_ibm_i_73/apifinder/apifinder40.htm
- the api finder page has +/- 5000 Api's
- System API programming manual
- Many examples on the web

www.midrange.com - www.think400.dk - www.scottklement.com

Google is your friend…

IFS API

- Example 1 : PURGEDIR
  - Delete all stream files older than x days in one directory and all sub directories.

  Featured IFS Api's from the C Library:

  - Opendir() Readdir() closedir() stat()
- Example 2 : COPY a source file to the IFS

  - create a .txt ascii stream file to the IFS.

  Open write close

User space

- A user space is a basic i5/OS object.
- It can be created and accessed with API
  - QUSCRTUS
  - QUSRTVUS
  - QUSCHGUS
  - QUSDLTUS
  - QUSPTRUS returns a pointer to the content of the userspace. So it is very fast…

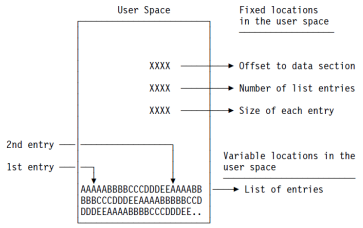
User Spaces

- Many list API can store data in user spaces
- We can access them with pointers

List Space format

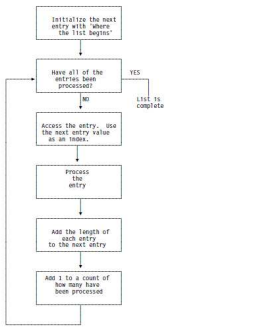
User Space Format—Example

Following is an example of the format of a user space. This example does not contain all of the fields in the fixed portion of a user space.



Logic Flow of Processing a List of Entries

When you process a list containing multiple entries, the logic flow looks as follows:



It is important from an upward compatibility viewpoint to use the offset, length of each entry, and the number of entries rather than hard coding the values in your program.

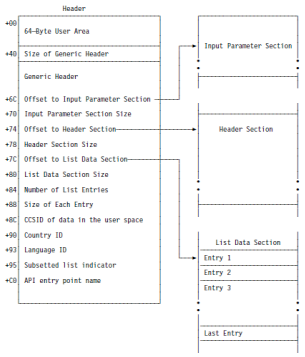


Figure 5-1. General Data Structure

List API

List Objects That Adopt Owner Authority API—Example

Parameters

Required Parameter Group:

1	Qualified user space name	Input	Char(20)
2	Format name	Input	Char(8)
3	User	Input	Char(10)
4	Object type	Input	Char(10)
5	Continuation handle	Input	Char(20)
6	Error code	I/O	Char(*)

The List Objects That Adopt Owner Authority (QSYLOBJP) API puts a list of objects that adopt an object owner's authority into a user space.

This API provides information similar to that provided by the Display Program Adopt (DSPPGMADP) command.

User Space Variables

The following tables describe the order and format of the data returned in the user space. For detailed descriptions of the fields in the tables, see "Field Descriptions" on page 5-15.

Input Parameter Section

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	User space name specified
10	0A	CHAR(10)	User space library name specified
20	14	CHAR(8)	Format name
28	1C	CHAR(10)	User name specified
38	26	CHAR(10)	Object type
48	30	CHAR(20)	Continuation handle 4

Header Section

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	User name
10	0A	CHAR(20)	Continuation handle 2

OBJP0100 Format

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Object name
10	0A	CHAR(10)	Library name
20	14	CHAR(10)	Object type
30	1E	CHAR(1)	Object in use

OBJP0200 Format

Offset			
Dec	Hex	Type	Field
0	0	CHAR(10)	Object name
10	0A	CHAR(10)	Library name
20	14	CHAR(10)	Object type
30	1E	CHAR(1)	Object in use
31	1F	CHAR(10)	Attribute
41	29	CHAR(50)	Text description

User space

- Example 3

  Get system status information (QUSRTVUS)
- Example 4
  - change library owner.
- Example 5
  - List spooled files

Process Open List API

- Asynchronous list processing
- These i5/OS list APIs can improve perceived performance when they create lists.
- The APIs create and make available to the caller a partial listing of the total set of files, messages, or objects. This list is immediately available to be acted upon, while the remainder of the list is being created. The user does not have to wait for the entire list to be created. Following is a description of the APIs and how they work together.

Process Open List API

- these APIs builds a list of the appropriate type and returns the number of records requested by the caller of the API. Also returned from the list building program is a request handle associated with that particular list. This request handle can be used on subsequent calls to the Get List Entry (QGYGTLE) API to get more records from the list. The request handle is valid until the Close List (QGYCLST) API is used to close the list.

Process Open List API

The request handle is also used as input to the following APIs when you need to find a specific entry in the list:

- Find Entry Number in List (QGYFNDE) API, which returns the number of the entry in a list of information for a given key value. This API can be used with lists that have been created by either the QGYOLOBJ or QGYOLSPL API.
- Find Entry Number in Message List (QGYFNDME) API, which returns the number of the entry in the list of message information for a given key value. This API can be used with lists that have been created by either the QGYOLMSG or QGYOLJBL API.
- Find Field Numbers in List (QGYFNDF) API, which returns the number of the entry in a list of information and the value of that entry whenever the value of that field changes.

Example

- Spooled file management cleanup application
  - Tables with retention info by Output Queue/Printer File
  - CREATE TABLE PSYSOUTQ
    - PSOQNM CHAR(10)
    - PSOQLB CHAR(10)
    - PSDAYS DEC(5 , 0))
  - CREATE TABLE PSYSPRTF
    - PSOQNM CHAR(10)
    - PSOQLB CHAR(10)
    - PSPRTF CHAR(10)
    - PRDAYS DEC(5 , 0)

Example

- List all outq in a userspace QUSLOBJ
- Read the user space
- For each entry /outq
  - Get retention days for the queue
  - Open a list of Spooled files
    - Check if known to get special retention/file
    - If expired : Run DLTSPLF
    - Get Next Entry
  - Close List

SQL

- Many system services are now available through SQL. (see SQL Update workshop)
- SQL is very popular
- Usually a very simpler access to the function than API programming
- See NETSTAT example

SQL services

```
SELECT *
    FROM
TABLE(QSYS2.OUTPUT_QUEUE_ENTRIES
('QGPL',    'QPRINT',   '*YES')) S
    ORDER BY SIZE DESC
```

MI Api

- Machine instructions are available as procedure in ILE environment
- All MI instructions can be inserted in ILE languages (COBOL, RPG, CL, C, …)
- Usually for a special system usage or a performance problem.

MI example

- MD5 calculation is used as a checksum to verify the integrity of data. (passwords, etc..)
- The MI instruction CIPHER provide the function…

Cipher (CIPHER)

Op Code (Hex)	Operand 1	Operand 2	Operand 3
10EF	Receiver	Controls	Source

Operand 1: Space pointer data object.

Operand 2: Character(32, 42, 96) variable scalar.

Operand 3: Space pointer data object.

Bound program access

```
Built-in number for CIPHER is 176.
CIPHER (
        receiver   : address of space pointer(16)
        controls   : address
        source     : address of space pointer(16)
)
```

Description: The cipher operation specified in the *controls* (operand 2) is performed on the string value addressed by the *source* (operand 3). The result is placed into the string addressed by the *receiver* (operand 1).

The *controls* operand must be a character variable scalar. It specifies information to be used to control the cipher operation. The common header of the *controls* operand has the following format.

Offset Dec	Offset Hex	Field Name	Data Type and Length
0	0	Function identifier	Char(2)

The *controls* operand must be a character variable scalar. It specifies information to be used to control the cipher operation. The common header of the *controls* operand has the following format.

Offset Dec	Offset Hex	Field Name	Data Type and Length
0	0	Function identifier	Char(2)
		The *function identifier* must be hex 0002, hex 0005, hex 0007, hex 0008, hex 0010, hex 0011, hex 0013, or hex 0015. If not, a *template value invalid* (hex 3801) exception is signaled.	
2	2	— End —	

The **function identifier** specifies the cryptographic service provider (CSP) for the cipher operation. It must specify hex 0002, hex 0005, hex 0007, hex 0008, hex 0010, hex 0011, hex 0013, or hex 0015. Any other value causes a *template value invalid* (hex 3801) exception to be signaled.

Hex 0002	The Machine CSP licensed internal code is to be used for a one-way encryption operation using the ANSI (American National Standards Institute) DEA (Data Encryption Algorithm).
Hex 0005	The Machine CSP licensed internal code is to be used to perform a one-way hash operation. The returned output may be a hash value or an HMAC (Hash Message Authentication Code) value. The supported hash algorithms are MD5 (Message Digest) and SHA-1 (Secure Hash Algorithm).
Hex 0007	The Machine CSP licensed internal code is to be used to perform a UNIX(R) crypt(3) operation.

Function Identifier 0005

The following description applies only to function identifier 0005.

The *controls* operand must be 16-byte aligned and have the following format:

Offset Dec	Offset Hex	Field Name	Data Type and Length
0	0	Controls operand	Char(96)
0	0		Function identifier Char(2)
2	2		Hash algorithm Char(1)
			Hex 00 = MD5
			Hex 01 = SHA-1
3	3		Sequence Char(1)
			Hex 00 = Only
			Hex 01 = First
			Hex 02 = Middle
			Hex 03 = Final
4	4		Data length UBin(4)
8	8		Output Char(1)