GRANITE SYSTEMS

Salutation Test Tool Command Reference and Workbook

Win32 version

February 15, 2000

Release Notes:

Release Date	Notes:
14-Nov-99	Initial Release • Remote extended fax event subscriptions are not supported. • Extended fax commands that utilize PSTN functions or derive from data transmission over fax lines for status updates or data transfers are not supported. Some of these commands are supported by the client FU but not the device emulation FU's. • PrnDsPaperInputTrayStatus command is not supported. • There is no graphical user interface for activation of commands. All commands must be invoked with the FU command line.
6-Dec-99	Update Release • Added references to VxWorks operating environment • Added notes regarding use of test tool in VxWorks environment • Added VxWorks testing column to all test grids • Clarified memory usage in FUMaster section • Clarified minimum parameter test of RmtPrint, RmtSendSfx, and RmtSendEfx • Deleted two redundant tests (test C in RmtGetPrivateAttribute and test B in RmtsetPrivateAttribute) • Reworded various command descriptions and test notes for clarity. • Added '–section' parameter to TodSendSfxTsinfoSet command • Completed testing of EfxChangePassword and RmtChangePassword • Completed testing of immediate mode 3rd party data transfers • Updated Efx/Sfx/QueryReadInformation/Rmt send commands to require todHandle parameter. • Updated test grid notes regarding testing of attributes for attribute types that do not exist in the Salutation definition.
15-Feb-00	Updated Release • Removed references to VxWorks • Removed all test grids •

Table of Contents

Introduction

Overview

This test tool is intended to assist in development of Salutation products. It assumes a high level of familiarity and understanding of the Salutation Level 1 and Level 2 architecture. It is capable of producing almost all Salutation commands that a Client FU might generate, and it will emulate (to various degrees) the responses and behaviors of print, docStore, fax, and extended fax FU's. The tool has a limited graphical interface, but at its core, it is a command line driven program. All commands may be issued from the textual command line, even those which would be considered inappropriate for a particular FU.

Where possible, the layout of commands exactly mirrors the corresponding Salutation Functional unit specification. Command syntax is taken directly from that specification, as are parameter names. The values of the many enumerations, (dynamic status Id's, attributes, etc) must be taken from the specification - most are not provided in this document, nor are they provided by the tool itself. In short, if you are using this tool, you will need a copy of the Salutation Architecture Specification readily available.

Local Commands vs. Remote Commands

Care should be taken to distinguish local FU commands (those which perform actions on the local client or device) and remote FU commands (those which issue commands across the salutation interface). Any command with a 'Rmt' prefix is a remote command - it will perform an FU level function on the remote FU which connected on the current session. Except for Session and SLM operations, all other commands are local commands - they affect the local FU by creating something, changing something, or performing some other operation.

Caution !!

There are hundreds of enumerated fields and other parameters used in the Salutation architecture. The valid uses, values, and combinations of these fields are documented in the architecture specification. Some of the enumerations are listed in the command descriptions for the user's convenience, but the Salutation Architecture Specification (part a, b and c) is the primary reference for the use of these commands and their parameterizations.

Because this is a test tool, both valid and invalid values may be placed in many command fields. Most local commands (those without the 'Rmt' prefix) operate without type or bounds checking, so it is possible to place the local FU in an inconsistent state through accidental or deliberate mistakes.

Remote commands may be transmitted with invalid values, or invalid commands issued by a user to a remote functional unit. This invalid command will usually be rejected by the remote FU (Nack'ed) and should cause no harm to the state of the local system. Because the test tool must allow a broad range of user actions, including many illegal actions, it may be induced to failure even by issuance of remote commands.

Example :

It is possible to send a 'print' command which specifies neither a data object definition (DOD) and without specifying a (third party) dataSource. In this case, the test tool will attempt to send a 'print' command without specifying the actual print data in any of the several possible ways which it can be specified. This is both architecturally invalid and a program error which will lead to unpredictable program results.

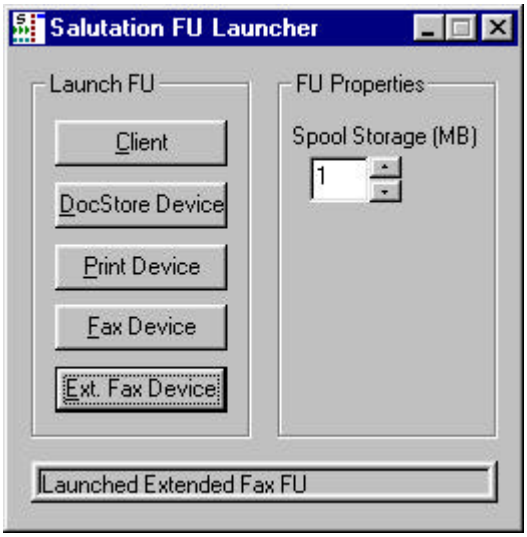
Some of these situations have been detected and a warning will be made in the 'notes' section of the appropriate command. The user should be aware that it is possible to accidentally or intentionally create a test program failure if they perform certain commands or command sequences with inconsistent parameterizations.

Installing and Running the Test Tool

The test tool requires no special installation, but there are several executable (.exe) a large number of dynamic link libraries (.dll's) required. Other than standard windows libraries, the test tool needs access to the specific sfu, test, and gui libraries, and also to certain slm and ber libraries.

FUMaster (Win32)

The Functional Unit Master program is a simple launcher utility for the functional units. It is invoked by executing the 'fumaster.exe' program. Each of the five supported test tool FU emulations may be started from this program (instead of from the command line). The launcher allows the user to specify the maximum amount of spool or data storage that an FU will use. All FU's require spool storage - even client FU's (for storage of data prior to sending jobs or documents to remote FU's).



A maximum of eight FU's may run at one time in one system, though there is no limitation on the total number of FU's which may run on network, as long as no more than eight are running on any one node. While it is possible to perform some operations on FU's with no spool storage (0 Mbytes), at least one Mbyte of storage is highly recommended. Spool sizes larger than 999MBytes may be specified through a manual entry in the size field.

GenFU (Win32)

The general FU program is invoked in one of five emulation mode : client, printer, document storage, standard fax, or extended fax. The type of FU to emulate is specified by clicking the appropriate button in the launcher. Be sure to first configure the amount of spool storage that the FU will support before launching the FU (this is not dynamically configurable once the FU is launched.

Alternatively, functional unit emulations may be invoked directly by executing the 'genfu.exe' program with command line arguments specifying the desired emulation mode, the requested fuHandle, and the total spool storage (in Mb).

genfu [-clt | -prn | -doc | -sfx | -efx] [-spoolStorage=<megabytes>] [-reqHandle=<int>]

-clt	client functional unit emulation
-prn	print functional unit emulation
-doc	docStore functional unit emulation
-sfx	send fax functional unit emulation
-efx	(extended) fax functional unit emulation

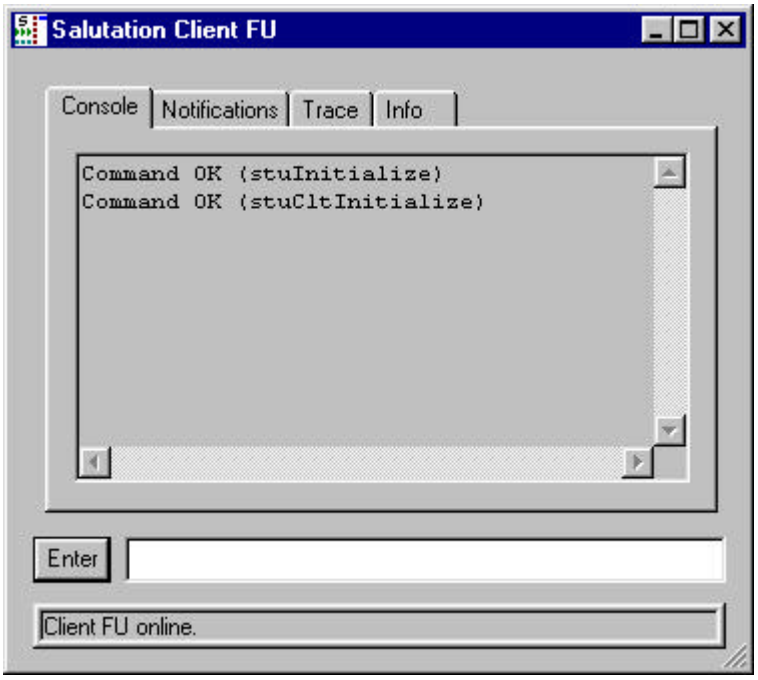
-spoolStorage=megabytes total data object storage area for the functional unit

-reqHandle=int requested handle for functional unit. If not a duplicate on the SLM, the FU will be created with this handle.

All FU's have four tabbed window pages which are used to provide the user with information. Also, there is a command line entry field which is used to enter commands to the FU.

Console Window

This is the dialog (input / output) window where most activity will take place. Test tool commands which are input on the command line are echoed in this window. Output from these commands is displayed in this window.



Notifications Window

Asynchronous notifications from event or job status subscriptions will be output in this window. When a remote (server) FU sends a notifyEvent or notifyJobStatus command to the local FU, the notification will be posted in this window.

Trace Window

A high-level message trace of the Salutation protocol exchange is output to this window. This is not a detailed trace, but it does identify the name of the messages and the order in which they are received and transmitted. See Trace Commands for syntax and usage.

Info Window

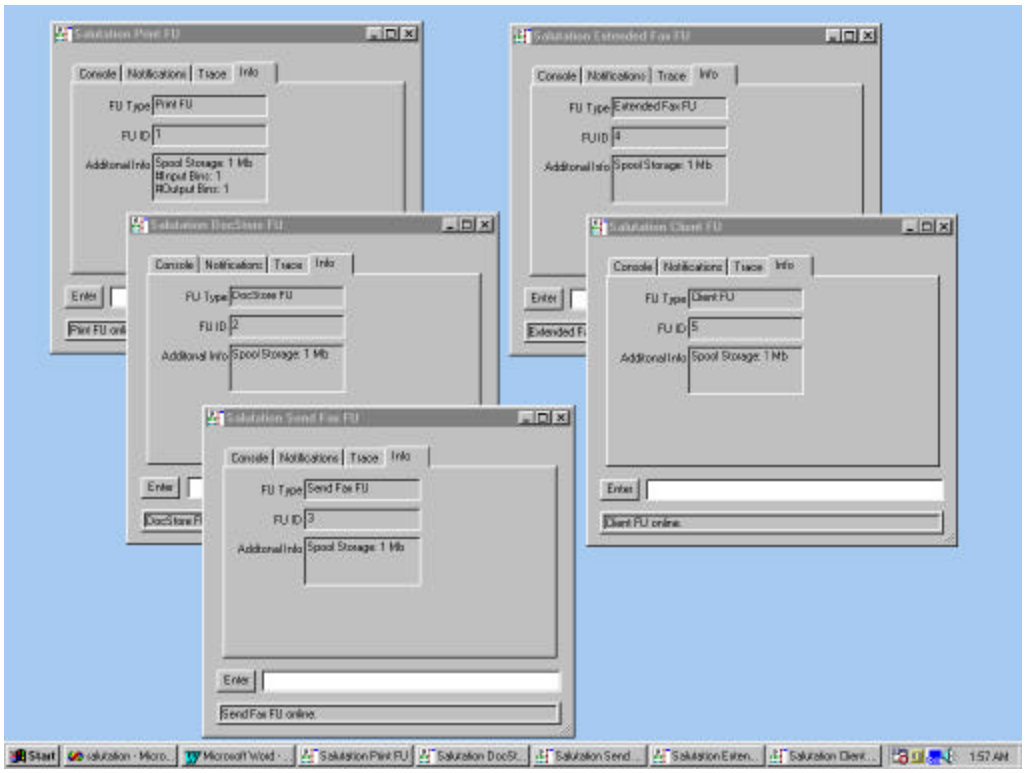
This window will contain important information about the FU, including its fuHandle, and its total spool storage.

Command Line Entry Field

Commands to the functional unit are typed in this field. To execute the command, press the enter button.

Reference Test Environment

Though customer test environments will vary significantly, much of the test tool functionality can be demonstrated with only test tool functional units. The SLM commands require at least two Win32 platforms to demonstrate all supported functionality - specifically, discovery of remote SLM's and their registered FU's, and retrieval of capability attributes registered for those FU's. In most circumstances, FU level functionality can be adequately demonstrated by running all supported FU's simultaneously on the same SLM - and thus on the same workstation. Since the only supported mode of SLM communications is TCP/IP based, using the loopback address (127.0.0.1) to establish and utilize sessions on the same SLM is sufficient for most testing needs.



If a customer has apprehensions about this mode of testing, or is looking to evaluate factors which require SLM to SLM communications, then these FU's may be run on separate workstations (with separate SLM's). The test tool itself was tested primarily with all FU's running on the same SLM, and spot checked (usually with data transfer commands) with FU's running on different SLM/workstations.

Command Syntax

The command syntax for test tool command line operation is illustrated in the example below.

SessionOpen ***-slmIp=<string>*** ***-fuHandle=<int>*** *-user={<string>$_{userId}$ <string>$_{password}$}$_{opt}$*

Command names are in **bold text** : **SessionOpen**

Required parameters are in ***bold-italic text*** : ***slmIp***

Optional parameters are in *italic text* : *user*

Parameter keys are in $_{italic\ subscript\ text}$: $_{password}$

(keys are not entered at the command line; they are simply an aid to the user)

Parameters, aggregates and lists are to be separated with at least one space character. Command line entries are not case sensitive. Neither flags nor field names are to be used inside aggregate and list structures.

Acceptable values and representation of values is summarized below.

int	*= 0..N or 0x0..N (hex format)*
enum	*= 0..N or 0x0..N (hex format)*
bool	*= true | false*
string	*= "value" (quotes are optional if no white space is included in the string)*
aggregate	*= {field1 … fieldN} (spaces between fields)*
lists	*= (value0 value1 value2 … valueN) (spaces between fields)*
IP address (string)	*= 111.222.333.444 (decimal dot format)*

An invocation of the SessionOpen command example listed above (with all options set) could look like this:

```
SessionOpen -slmIp=127.0.0.1 -fuHandle=1 -user={kqlee mypassword}
```

Optional commands do not need to be included. The rules for the use of optional commands are described in the Salutation FU specifications. In this case, the above command could also be invoked as:

```
SessionOpen -slmIp=127.0.0.1 -fuHandle=1
```

Glossary of Acronyms

Item	Definition
Doc	Document Storage (FU)
Dod	Data Object Definition
Ds	Dynamic Status
Efq	Extended Fax Queue
Efx	Extended Fax (FU)
Fsi	Fax Send Info
FU	Functional Unit
Rcoi	Read Confirmation Order Info
Rmt	Remote
Sdi	Send Destination Info
Sfx	Send Fax (FU)
Slm	Salutation Manager Level
Sui	Send User Info
Tod	Transfer Object Definition

Control Commands

These are functions which exceed the scope of the normal command set because they affect or expose the operation of the interpreter itself, like exiting, reading from a script file, or providing online help.

quit	Exit from an FU test tool
help	List available (enabled, non-hidden) commands, or give detailed help on a given command
echo	Print out a user-defined string
script	Run a text file script of commands

Quit

syntax :

Quit

description :

Exit from the FU test tool. Free all allocated resources and unregister the FU from the local SLM. The local SLM will remain active.

result :

N/A

example :

```
>Quit
```

Help

syntax :

Help *-cmd=<string>$_{opt}$ -usage=<bool>$_{opt}$*

description :

Display help, parameter type and usage syntax, for the specified command. Using this command without the –cmd parameter will provide a listing of all available commands.

parameters:

Parameter	Description
-cmd	A valid test tool command. If omitted, a listing of all commands.
-usage	Boolean value determining if usage should be listed. If omitted, usage will be displayed (True).

result :

Special – listing of all commands, command parameters, or command parameters and usage.

example :

```
> help -cmd=help -usage=true;
usage: help [-cmd=<string>] [-usage={true|false}]
```

Echo

syntax :

Echo *-str=<string>*$_{opt}$

description :

Echo the specified string to the command-line dialog window.

parameters:

Parameter	Description
-str	String that will be displayed in the FU Console window. String must be enclosed by quotes if any white-space is used in the string.

result :

Special – text displayed in console

example :

```
> Echo -str="hello world";
hello world
```

Script

syntax :

Script *–**scr**=<**filename**>* *–errorContinue=<bool>*$_{opt}$

description :

Invoke a script command file. Included with the test tool are example scripts. Script files allow the user of the tool to execute a sequence of commands as if they were entered on the command line. Scripts can be invoked from the command line or from the GUI tabbed page. The text file scripts may be identified by a '.txt' filename suffix. This suffix is purely convention, it is not required for operation of the test tool - any suffix or no filename suffix may be used. Using scripts as an example is probably the easiest way to get started using the test tool.

Script files must be saved as ASCII text files. The "#" character is used for beginning comments; any text on a line after the "#" will be ignored by the command interpreter. Each command must end with ";" (semicolon) to signify the end of the command. The command interpreter will ignore CR and extra space characters. Commands and parameters are not case-sensitive.

Shown below is a brief example script that creates a data object and then lists details about the data object.

```
#######################################################################
#                            Example Script                           #
#######################################################################

## Create a DOD from a valid file.
DodCreateFromFile            # Command can be on several lines
 -dodHandle=10               #
 -filename=file.txt;         # be sure to end command with ";"

## Create a DOD from a file that does not exist.
DodShow -dodHandle=10;       # Command can be on one line
```

parameters:

Parameter	Description
-scr	File name of the script file. A fully qualified DOS path can be included as a part of the file name. If no DOS path is defined, the file is assumed to be in the same directory as where the FUMaster.exe file was launched.
-errorContinue	Continue script on Command ERROR == True Stop on Command ERROR == False The Script will always abort for syntax errors or other processing type errors.

result :

Special - displays a run-time dialog of scripting output

example :

```
script -scr=startup.txt –errorContinue=true;
```

SLM Commands

These several functions provide a direct interface into the Salutation Manager level of functionality. They allow a user to determine what Managers are available, what FU's are resident on those managers, and what the capabilities of those FU's are.

Functional Unit Name	ID
Wild (for use with QueryCapability)	0
[Client]	1000
[Print]	10000
[Document Storage]	11000
[FAX Data Send]	12000
[FAX Data]	13000
[Voice Message Storage]	20000
[Address Book]	30000

The following are the available Salutation Manager commands:

SlmList
SlmListFU
SlmListCap

SlmList

syntax :

SlmList

description :

This command lists the active salutation managers on this network segment.

result :

Command OK : List of SLM's on network segment by SLMID. The local SLMID reported as all zeroes.
Command ERROR

notes :

Only TCP/IP transport for SLM communications is supported.

example :

```
> slmList;
num slmid = 2
slmid type = 5 , data = 0xA 0x0 0x0 0x64 0x86 0xF2 0x34 0x36 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0
slmid type = 0 , data = 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0
Command OK (stuSlmList)
```

SlmListFU

syntax :

SlmListFU *-slmIp=<string$_{ipaddress}$>*

description :

List all FU's registered with the specified Salutation Manager.

parameters:

Parameter	Description
-slmIp	Internet address of remote SLM in dot-decimal or symbolic format

result :

Command OK - List of FU (handle and id) on remote or local SLM.
Command ERROR

notes :

This command will also list FU's registered on the local SLM (if used with the local or loopback IP address). If you are using Windows Dial-Up Networking, Windows may attempt to dial out as it searches for the specified SLM IP. If the dialer starts, cancel it and Windows will continue to search on the local network.

The test tool does not support an SLM with no FU active. In an application where an SLM exists with no FU active, the local FU results will be “Command OK - empty list displayed”

example :

```
> slmlistfu -slmip=127.255.255.10;
FU Id = 13000 , Handle = 244193640
FU Id = 10000 , Handle = 1315525413
FU Id = 11000 , Handle = 1100510003
FU Id = 1000 , Handle = 1872038571
FU Id = 12000 , Handle = 1
Command OK (stuSlmListFU)
```

SlmListCap

syntax :

SlmListCap ***-slmIp=<string$_{ipaddress}$>*** *-fuHandle=<int>$_{opt}$* *-fuId=<int>$_{opt}$*

description :

List (in decoded format) the Id's and values of capability attributes of the specified FU(s). FU's may be specified by their unique handle, and/or by their type id.

parameters:

Parameter	Description
-slmIp	Internet address of remote or local SLM in dot-decimal or symbolic notation. Using this parameter alone will create a listing of all FU's on the SLM.
-fuHandle	SLM unique handle assigned to FU upon registration. Use this parameter to limit the listing to one particular FU
-fuId	Type of FU (defined in Salutation Architecture Specification, Part-2). Use this parameter to limit the listing to one particular FU type. All FU's of that type will be listed.

result :

Command OK : list of FU's and decoding of their capability attributes
Command ERROR

notes :

This command will also list capability attributes of FU's registered on the local SLM (if used with the local or loopback IP address). If you are using Windows Dial-Up Networking, Windows may attempt to dial out as it searches for the specified SLM IP. If the dialer starts, cancel it and Windows will continue to search on the local network.

example :

```
> slmListCap -slmIp=127.0.0.1 -fuHandle=1;
FU Id = 1000 , Handle = 1
attributeId=10 , compare=5 int-value = 2
attributeId=11 , compare=5 int-value = 0
attributeId=20 , compare=5 int-value = 17
attributeId=30 , compare=5 octet-string-value = Granite
attributeId=40 , compare=5 octet-string-value = IBM
attributeId=41 , compare=5 octet-string-value = Jake
attributeId=42 , compare=5 octet-string-value = Ver1.0
attributeId=50 , compare=5 octet-string-value = 100 Main St., Longmont CO 80501
attributeId=51 , compare=5 octet-string-value = kqlee@gsco.com
attributeId=60 , compare=5 set-value(s) = 0 1
attributeId=1000 , compare=5 octet-string-value = kqlee
Command OK (stuSlmListCap)
```

Session Commands

All FU level communication takes place on a Salutation session which must be explicitly opened between two FU's. Multiple sessions may be opened between the same or different FU's. The session functions allow for the creation and management of salutation sessions by the test tool.

Before remote commands are issued, a session must be opened between the local FU and a remote FU. The most recently opened (or selected) session is implicitly selected for the transmission of remote commands. Several sessions may be opened at one time. However, only one session at a time is available for issuing remote commands. The session that is currently selected for transmission is the 'current session'.

The following are the available session commands:

- SessionOpen
- SessionClose
- SessionSelect
- SessionListShow
- SessionShow

SessionOpen

syntax :

SessionOpen ***-slmIp=<string$_{ipaddress}$>*** ***-fuHandle=<int>*** *-user={<string$_{userId}$> <string$_{password}$>}$_{opt}$*

description :

The SessionOpen command opens a session to the specified FU. User/password authentication is optional. If the user/password is not specified, then the session is opened without authentication. Subsequent FU transactions will be conducted on this session, which is set as the currently selected session (until another session is opened or selected as the active session).

parameters:

Parameter	Description
-slmIp	Internet address of remote SLM in dot-decimal or symbolic format
-fuHandle	SLM unique handle assigned to FU upon registration
-user	UserID / Password for session authentication

result :

Command OK
Command ERROR

notes :

1. An FU should not open a session to itself. The results for this action are undefined.
2. The remote FU to which the session has been created will not automatically select the new session as its current session. You will need to go to the remote FU and do a SessionSelect command.
3. To open an “Administrator” session, open the session with “Administrator” as the user and any password. Some remote Efx commands require an administrator session.
4. Register a user on an Extended Fax FU. You must have a current authenticated
5. If you are using Windows Dial-Up Networking, Windows may attempt to dial out as it searches for the specified SLM IP. If the dialer starts, cancel it and Windows will continue to search on the local network.

example :

```
> SessionOpen -slmIp=127.0.0.1 -fuHandle=1 -user={ kqlee mypassword };
svc handle = 5262336
Command OK (stuSessionOpen)
```

SessionClose

syntax :

SessionClose *-sessionHandle=<int>*$_{opt}$

description :

Close the specified session. If no session Id is specified, close the current open session. A new current session is not automatically selected.

parameters:

Parameter	Description
-sessionHandle	FU unique handle assigned to session when it is opened on local FU

result :

Command OK
Command ERROR

notes :

You will need to perform a SessionSelect command following a SessionClose command to select a new current session.

example :

```
> SessionClose -sessionHandle=5262336;
Command OK (stuSessionClose)
```

SessionSelect

syntax :

SessionSelect *-sessionHandle=<int>*

description :

Select a different current session. Remote transactions are conducted on the current session, so changing this value will redirect subsequent remote commands.

parameters:

Parameter	Description
-sessionHandle	FU unique handle assigned to session when it is opened on local FU

result :

Command OK

notes :

This command does not check that the newly assigned current session id is valid.

example :

```
> SessionSelect -sessionHandle=5262337;
Command OK (stuSessionSelect)
```

SessionListShow

syntax :

SessionListShow

description :

List the sessionHandle of all open sessions on this FU. Also lists the current session.

result :

Command OK : List of session handles (possibly empty) and current session

example :

```
SVC_LIST (handles) : 4405860 4404084
Current Session = 4404084
Command OK (stuSessionListShow)
```

SessionShow

syntax :

SessionShow *-sessionHandle=<int>*

description :

Show details of an open session.

parameters:

Parameter	Description
-sessionHandle	FU unique handle assigned to session when it is opened on local FU

result :

Command OK : Details of open session
Command ERROR

example :

```
> SessionShow -sessionHandle=5262336;
 sessionHandle : 5262336
 rmt.slmid     : 5:c0.9.c8.c9.f9.6c.ed.34.0.0.0.0.0.0.0
 rmt.fuhandle      : 1
 personality   : 0
 PATT_LIST (handles) :  999  998
Command OK (stuSessionShow)
```

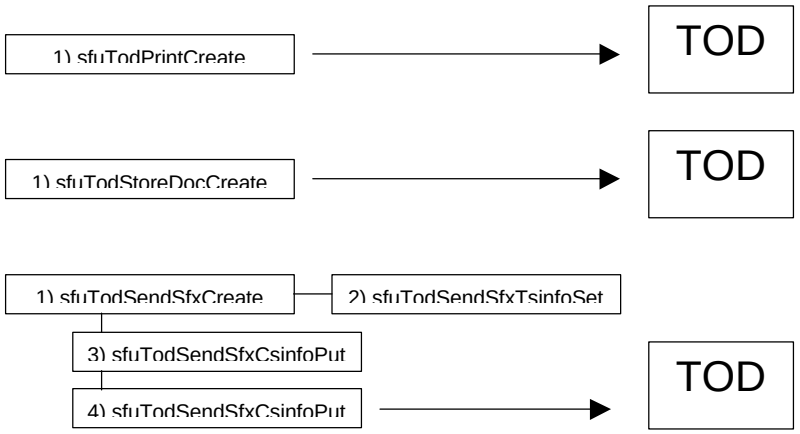
Transfer Object Definition Commands (Tod)

TOD Overview

Commands which involve data transfers to a remote FU may use a transfer object definition (TOD) as a template for execution. Similarly, commands which queue jobs directly on a local FU (or store documents) may use the TOD. The motivation for the TOD based interface is rooted in the deep nesting and complexity of some job description formats - especially the Extended Fax FU. Using a TOD is optional for a Print and StoreDoc command - since they have no nested components and can be completely specified in a single command. The SendFax, SendExtFax, and QueryReadInformation commands must first build a TOD, since that is the only way to specify the complicated nested and optional features of these commands. Once defined, a TOD is persistent in the test tool space and may be reused any number of times until it is explicitly destroyed.

Construction of TOD's

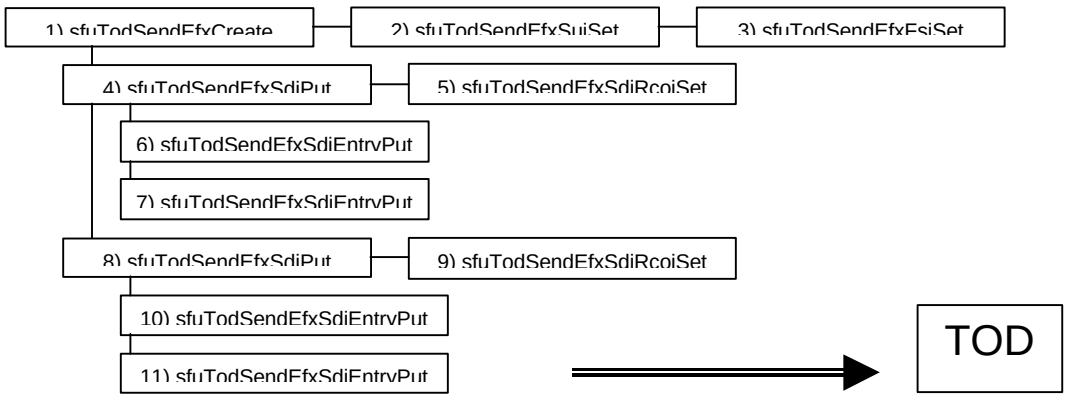
Constructing a TOD must be done in a top down fashion. The basic TodXXXCreate call is first made, and then successively deeper nested structures may be set or put. A command with a 'set' suffix indicates that a single instance of the parameter values are recorded at that level. A command with a 'put' suffix indicates that multiple instances of the declared sub-object are recorded at that level (for example, job entries).



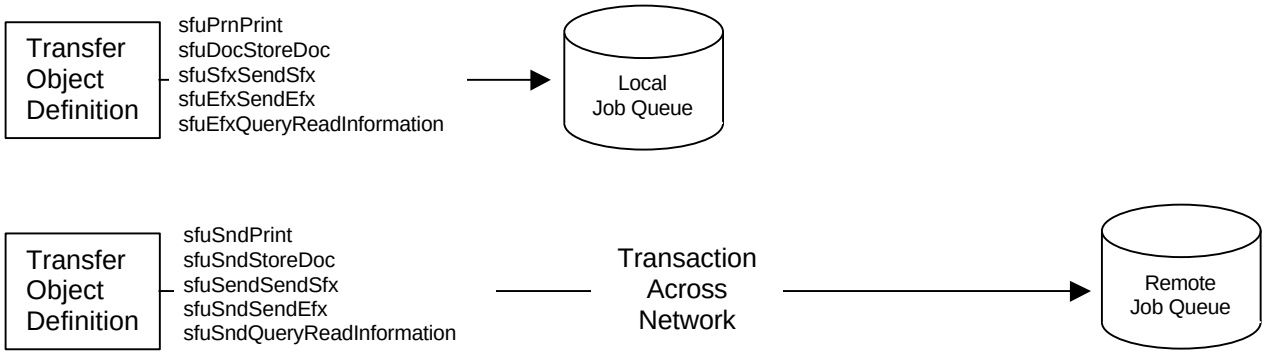
Both Print TOD's and StoreDoc TOD's can be created in a single command.

SendFax TOD's must be first created, then TsInfo can be set, and entries added to the definition.

Extended Fax (Efx) TOD's must fist be created, then the Sui and Fsi structures can be set, and Sdi's added. Once Sdi's are added, then their Rcoi structures can be set and entries added to them.

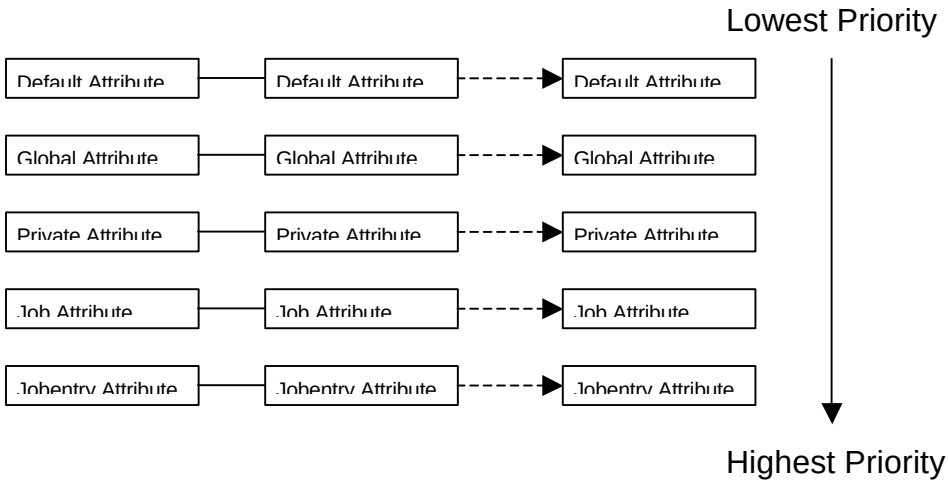


Since TOD's are used to define objects in preparation of remote queuing (like a client FU would do) and local queuing (such as a prn/fax FU would do), they do not perform bounds checking on supplied parameters. The TodChk command will verify the value of the TOD and its sub-components against the local FU's capability attribute values. This should not be done by client FU's, since the relevant capability attributes are on the remote FU (the target of the print/storeDoc/sendFax… operation).



Local Queuing Operations from TOD's

When local queuing operations are conducted (sfuPrnPrint, sfuSfxSendSfx, sfuEfxSendEfx, sfuEfxQueryReadInformation) from TOD's (or in several cases, no TOD is necessary), a record is made of the current attribute state for the operation. This is more than is required by the Salutation Architecture, but is often useful to clients of the job queue. First, all default attributes are recorded in a list, these are added to or overridden by the set of current FU global attributes, then added to or overridden by the appropriate session 'private' attributes, and finally the recorded job attributes or jobentry attributes from the definition record are added to the list or override lower priority attribute value settings. In the end, the job or entry holds a list of the complete and current attribute state at the time of its queuing.



The attribute state is available through remote and local job attribute commands which are supported by the test tool SDK. It can be examined (again exceeding the specifications of the SDK through the 'rmtGetJobAttribute' command.

Tod's are maintained in a list on the FU until they are explicitly destroyed.

The following are the available Tod commands:

- TodDestroy
- TodStoreDocCreate
- TodPrintCreate
- TodSendSfxCreate
- TodSendSfxTsinfoSet
- TodSendSfxCsinfoPut
- TodQueryReadInformationCreate
- TodQueryReadInformationEfqPut
- TodQueryReadInformationEfqEntryPut
- TodSendEfxCreate
- TodSendEfxSuiSet
- TodSendEfxFsiSet
- TodSendEfxSdiPut
- TodSendEfxSdiRcoiSet
- TodSendEfxSdiEntryPut
- TodDodSet
- TodListShow
- TodShow

TodDestroy

syntax :

TodDestroy *-todHandle=<int>*

description :

Destroy a transfer object definition.

parameters:

Parameter	Description
-todHandle	FU unique handle assigned to TOD when it was created

result :

Command OK
Command ERROR

notes :

Destroying a TOD will automatically decrement reference count of any referenced data object definition (DOD) and may result in the destruction of that DOD (if this is the only remaining reference).

example :

```
> TodDestroy -todHandle=3;
Command OK (stuTodDestroy)
```

TodStoreDocCreate

syntax :

TodStoreDocCreate *-todHandle=<int> -dodHandle=<int> -folderId=<int>*
-dataSource={<ipaddress$_{slmIp}$><int$_{fuHandle}$>}$_{opt}$ -dataHandle=<int>$_{opt}$ -modeOfStore=<enum>$_{opt}$
-ownerName=<string>$_{opt}$ -docComment=<string>$_{opt}$

description :

Create a new transfer object definition for a 'DocStoreDoc' or 'RmtStoreDoc' command.

parameters:

Parameter	Description
-todHandle	FU unique handle assigned to TOD by this command
-dodHandle	FU unique handle assigned to DOD when it was created. Including a dodHandle will create an additional reference to that DOD.
-folderId	Default folder Id where document is to be placed
-dataSource	Default dataSource from which data is to be transferred
-dataHandle	Default handle of data to be transferred for document
-modeOfStore	Default mode of store 1 == Non-transparent Mode - transmit and store data as multiple blocks (retain block boundaries as when created) 2 == Transparent Mode - transmit and store data a single block (combine blocks)
-ownerName	Default document ownerName
-docComment	Default document comment

result :

Command OK
Command ERROR

notes :

1) TodHandles above 0x1000 (4096) are reserved for system use. Do not assign values above this value.
2) No parameter range and bound checking is performed by this command.
3) The DataSource must be resolvable into an actual SLMID at the time this TOD is created. The command will search for the ID based on the supplied IP address - if it is not found, then the command will return an error.

example :

```
> TodStoreDocCreate -todHandle=5 -dodHandle=1 -folderId=2;
Command OK (stuTodStoreDocCreate)
```

TodPrintCreate

syntax :

TodPrintCreate ***-todHandle=<int>*** *-dodHandle=<int>$_{opt}$ -modeOfDataTransfer=<enum>$_{opt}$ -dataSource={<ipaddress$_{slmIp}$> <int$_{fuHandle}$>}$_{opt}$ -dataHandle=<int>$_{opt}$ -life=<enum>$_{opt}$ -notificationMode={(<enumlist$_{jobStatusCode}$>) <bool$_{entries}$>}$_{opt}$ -notificationScheme={<ipaddress$_{slmIp}$><int$_{fuHandle}$>}$_{opt}$ -paperSize=<enum>$_{opt}$ -resolution=<enum>$_{opt}$ -paperDirection=<enum>$_{opt}$ -copyCount=<int>$_{opt}$ -inputSelect=<enum>$_{opt}$ -outputSelect=<enum>$_{opt}$ -outputBinSelect=<int>$_{opt}$ -duplexModeSelect=<enum>$_{opt}$ -duplexBindingMargin=<int>$_{opt}$ -faceUpModeSelect=<enum>$_{opt}$ -priority=<int>$_{opt}$ -staplingSelect=<enum>$_{opt}$ -fileName=<string>$_{opt}$*

description :

Create a transfer object definition for a 'PrnPrint' or 'RmtPrint' command.

parameters:

Parameter	Description
-todHandle	FU unique handle assigned to TOD by this command
-dodHandle	FU unique handle assigned to DOD when it is created. Including a dodHandle will create an additional reference to that DOD.
-modeOfDataTransfer	Enumerated list – immediate == 0, delayed == 1. If not defined, immediate is assumed.
-dataSource	Default source of data for transfer. Must include IP address of SLM and FU ID of the data source.
-dataHandle	Default handle of data to transfer
-life	Default job life. Enumerated list – job == (0), session == (1), persistent == (2)
-notificationMode	Default notification mode. Parameter includes a set of Job Status Codes and a boolean value allowing for a request for Job Entry Status. Only a Client FU can receive job status notifications. Enumerated list – see Salutation Architecture Specification, V2.0, Part 2, Page 201 for a listing of job status codes.
-notificationScheme	Default notification scheme. This parameter allows for notification to an FU other than the one that has initiated the job. Only a Client FU can receive job status notifications.
-paperSize	Default paper size. Enumerated list – see Salutation Architecture Specification, V2.0, Part 2, Page 210.
-resolution	Default print resolution. Enumerated list – see Salutation Architecture Specification, V2.0, Part 2, Page 208
-paperDirection	Default paper direction. Enumerated list – portrait == (1), landscape == (2), other == (127)
-copyCount	Default copy count. Integer defining the number of copies requested
-inputSelect	Default paper tray input selection. Enumerated list - manualFeed == (0), tray-1 == (1), tray-2 == (2), tray-3 == (3), tray-4 == (4), tray-5 == (5), automaticSelect (FU selects tray by PaperSize) == (126), other == (127)
-outputSelect	Default output sortation mode. Enumerated list - standard == (0), collatedSort == (1), stack == (2), nonCollatedSort == (3), other == (127)
-outputBinSelect	Default output bin number. Integer corresponding to equipment bin number
-duplexModeSelect	Default duplex binding mode. Enumerated list - simplex == (0), left-binding-duplex == (1), right-binding-duplex == (2), top-binding-duplex == (3), other == (127)
-duplexBindingMargin	Margin distance for duplex binding. (0.1mm x duplexBindingMargin)??
-faceUpModeSelect	Default mode selection. Enumerated list – faceDown == (1), faceUp == (2), other == (127)
-priority	Default priority. Integer (0..100) -- 50 is the normal priority, -- 100 is the highest priority. Valid values are enumerated in the capability attributes on a per FU basis.
-staplingSelect	Default stapling location. Enumerated list – see Salutation Architecture Specification, V2.0, Part 2, Page 215.
-fileName	Default file name of print job

result :

Command OK
Command ERROR

notes :

TodHandles above 0x1000 (4096) are reserved for system use. Do not assign values above this value.

No parameter range and bound checking is performed by this command.

The DataSource-ipAddress and NotificationScheme-ipAddress must be resolvable into an actual SLMID at the time this TOD is created. The command will search for the ID based on the supplied IP address - if it is not found, then the command will return an error.

example :

```
> TodPrintCreate -todHandle=6 -life=1 -notificationMode={ ( 1 2 ) true } -filename="example.prn";
Command OK (stuTodPrintCreate)
```

TodSendSfxCreate

syntax :

TodSendSfxCreate ***-todHandle=<int>*** *-dodHandle=<int>$_{opt}$ -modeOfDataTransfer=<enum>$_{opt}$ -dataSource={<ipaddress$_{slmIp}$><int$_{fuHandle}$>}$_{opt}$ -dataHandle=<int>$_{opt}$ -life=<enum>$_{opt}$ -notificationMode={(<enumlist$_{jobStatusCode}$>) <bool$_{entries}$>}$_{opt}$ -notificationScheme={<ipaddress$_{slmIp}$><int$_{fuHandle}$>}$_{opt}$ -priority=<int>$_{opt}$ -retryCount=<int>$_{opt}$*

description :

Create a transfer object definition for a 'SfxSendSfx' or 'RmtSendSfx' command. This command declares parameters which are transmitted in the base SendFAX command, and the FaxControlAttribute structure. Sfx TOD's must be created with this command, then the TSInfo structure may be set (TodSendSfxTsinfoSet) and entries added to the base job (TodSendSfxCsinfoPut).

parameters:

Parameter	Description
-todHandle	FU unique handle assigned to TOD by this command
-dodHandle	FU unique handle assigned to DOD when it was created. Including a dodHandle will create an additional reference to that DOD.
-modeOfDataTransfer	immediate == 0 delayed == 1
-dataSource	Default source of data for transfer
-dataHandle	Default handle of data to transfer
-life	job == 0 session == 1 persistent == 2
-notificationMode	Default notification mode. Parameter includes a set of Job Status Codes and a boolean value allowing for a request for Job Entry Status. Only a Client FU can receive job status notifications. Enumerated list – see Salutation Architecture Specification, V2.0, Part 2, Page 201 for a listing of job status codes.
-notificationScheme	Default notification scheme. This parameter allows for notification to an FU other than the one that has initiated the job. Only a Client FU can receive job status notifications.
-priority	Default priority. Integer (0..100) -- 50 is the normal priority, -- 100 is the highest priority. Valid values are enumerated in the capability attributes on a per FU basis.
-retryCount	FU tries to re-call the remote telephone retryCount times, when the remote telephone was busy.

result :

Command OK
Command ERROR

notes :

TodHandles above 0x1000 (4096) are reserved for system use. Do not assign values above this value.

No parameter range and bound checking is performed by this command.

The DataSource-ipAddress and NotificationScheme-ipAddress must be resolvable into an actual SLMID at the time this TOD is created. The command will search for the ID based on the supplied IP address - if it is not found, then the command will return an error.

example :

```
> TodSendSfxCreate -todHandle=7 -notificationScheme={ 206.196.145.225  1 } -retryCount=3;
Command OK (stuTodSendSfxCreate)
```

TodSendSfxTsinfoSet

syntax :

TodSendSfxTsinfoSet ***-todHandle=<int>*** *-name=<string>$_{opt}$ -section=<string>$_{opt}$ -company=<string>$_{opt}$ -phoneNumber=<string>$_{opt}$ -faxNumber=<string>$_{opt}$ -address=<string>$_{opt}$ -subject=<string>$_{opt}$ -coverSheetGen=<bool>$_{opt}$ -memoForCover=<string>$_{opt}$ -pageHeaderGen=<bool>$_{opt}$ -memoForHeader=<string>$_{opt}$*

description :

Set the Tsinfo values for an sfx job transfer definition. The Tsinfo structure is part of the FaxControlAttribute structure. The entire Tsinfo structure is optional, so if this command is not used, then no Tsinfo is declared, and no Tsinfo structure is transmitted to a remote FU when a RmtSendSfx command is issued.

parameters:

Parameter	Description
-todHandle	FU unique handle assigned to a TOD created by TodSendSfxCreate
-name	String value for fax name field
-section	String value for fax section field
-company	String value for fax company field
-phoneNumber	TelephoneNumberString value for fax phone number field
-faxNumber	TelephoneNumberString value for fax number
-address	String value for fax address field
-subject	String value for fax subject field
-coverSheetGen	Boolean value
-memoForCover	String value for cover sheet memo field
-pageHeaderGen	Boolean value
-memoForHeader	String value for header memo field

result :

Command OK
Command Error

example :

```
> TodSendSfxCreate -todHandle=7 -notificationScheme={ 206.196.145.225  1 } -retryCount=3;
Command OK (stuTodSendSfxCreate)
> TodSendSfxTsinfoSet -todHandle=7 -name="Kerry Lee" -coverSheetGen=false;
Command OK (stuTodSendSfxTsinfoSet)
```

TodSendSfxCsinfoPut

syntax :

TodSendSfxCsinfoPut *-todHandle=<int> -jobEntryId=<int> -faxNumber=<string>*
-subAddressNumber=<string>$_{opt}$ -name=<string>$_{opt}$ -section=<string>$_{opt}$ -company=<string>$_{opt}$
-phoneNumber=<string>$_{opt}$ -address=<string>$_{opt}$ -faxProtocol=<enum>$_{opt}$ -orderingData=<string>$_{opt}$

description :

Put a job entry to an Sfx job transfer definition. This correlates to adding a Csinfo structure to the list in the FaxControlAttribute structure - which has the effect of adding a job entry to the fax job.

parameters:

Parameter	Description
-todHandle	FU unique handle assigned to a TOD created by TodSendSfxCreate
-jobEntryId	Job entry id assigned by user in this command
-faxNumber	TelephoneNumberString value for fax telephone number
-subAddressNumber	String value for subaddress number field
-name	String value for fax name field
-section	String value for fax section field
-company	String value for fax company field
-phoneNumber	TelephoneNumberString value for fax phone number field
-address	String value for fax address field
-faxProtocol	g3 == (1) G3 is assumed when omitted g4 == (2) auto selection == (3)
-orderingData	TelephoneNumberString value for final recipient telephone number

result :

Command OK
Command ERROR

notes :

This command performs no range or bound parameter checking.

example :

```
> TodSendSfxCsinfoPut -todHandle=7 -jobEntryId=1 -faxNumber="687-9876";
Command OK (stuTodSendSfxCsinfoPut)
```

TodQueryReadInformationCreate

syntax :

TodQueryReadInformationCreate ***-todHandle=<int>***
-notificationMode={(<enumlist$_{jobStatusCode}$>) <bool$_{entries}$>}$_{opt}$
-notificationScheme={<ipaddress$_{slmIp}$> <int$_{fuHandle}$>}$_{opt}$ -retryCount=<int>$_{opt}$ -retryInterval=<int>$_{opt}$
-scheduledDateTime=<string>$_{opt}$ -scheduledAfterTime=<string>$_{opt}$ -priority=<int>$_{opt}$

description :

Create a TOD for use in 'EfxQueryReadInformation' or 'RmtQueryReadInformation' command. This command should be followed by declaration of extended fax queries (TodQueryReadInformationEfqPut).

parameters:

Parameter	Description
-todHandle	FU unique handle assigned to TOD by this command
-notificationMode	Default notification mode. Parameter includes a set of Job Status Codes and a boolean value allowing for a request for Job Entry Status. Only a Client FU can receive job status notifications. Enumerated list – see Salutation Architecture Specification, V2.0, Part 2, Page 201 for a listing of job status codes.
-notificationScheme	Default notification scheme. This parameter allows for notification to an FU other than the one that has initiated the job. Only a Client FU can receive job status notifications.
-retryCount	FU tries to re-call the remote telephone retryCount times, when the remote telephone was busy.
-retryInterval	FU tries to re-call the remote telephone at retryInterval minute intervals.
-scheduledDateTime	Time for scheduled future delivery. Expressed in UTCTime format.
-scheduledAfterTime	Time delay before delivery. Expressed in "hh:mm:ss" format.
-priority	Default priority. Integer (0..100) -- 50 is the normal priority, -- 100 is the highest priority. Valid values are enumerated in the capability attributes on a per FU basis.

result :

Command OK
Command ERROR

notes :

TodHandles above 0x1000 (4096) are reserved for system use. Do not assign values above this value. No parameter range and bound checking is performed by this command.
The NotificationScheme-ipAddress must be resolvable into an actual SLMID at the time this TOD is created. The command will search for the ID based on the supplied IP address - if it is not found, then the command will return an error.

example :

```
> TodQueryReadInformationCreate -todHandle=8 -priority=50;
Command OK (stuTodQueryReadInformationCreate)
```

TodQueryReadInformationEfqPut

syntax :

TodQueryReadInformationEfqPut *-todHandle=<int> -sourceJobHandle=<int>*

description :

Add an Efx (Extended Fax) query to a transfer definition. This command should be followed by declarations of job entries (TodQueryReadInformationEfqEntryPut) for this extended fax query.

parameters:

Parameter	Description
-todHandle	FU unique handle assigned to a TOD created by TodQueryReadInformationCreate
-sourceJobHandle	Job handle (on remote FU) to be queried

result :

Command OK
Command ERROR

example :

> TodQueryReadInformationCreate -todHandle=8 -priority=50;
Command OK (stuTodQueryReadInformationCreate)
> TodQueryReadInformationEfqPut -todHandle=8 -sourceJobHandle=3;
Command OK (stuTodQueryReadInformationEfqPut)

TodQueryReadInformationEfqEntryPut

syntax :

TodQueryReadInformationJobEntryPut ***-todHandle=<int> -sourceJobHandle=<int> -jobEntryId=<int> -userId=<string>***

description :

Add a job entry info structure to an extended fax query of a transfer definition.

parameters:

Parameter	Description
-todHandle	FU unique handle assigned to a TOD created by TodQueryReadInformationCreate
-sourceJobHandle	Job handle (on remote FU) to be queried
-jobEntryId	Job entry id (on remote FU) to be queried
-userId	User to be queried

result :

Command OK
Command ERROR

example :

```
> TodQueryReadInformationCreate -todHandle=8 -priority=50;
Command OK (stuTodQueryReadInformationCreate)
> TodQueryReadInformationEfqPut -todHandle=8 -sourceJobHandle=3;
Command OK (stuTodQueryReadInformationEfqPut)
> TodQueryReadInformationEfqEntryPut -todHandle=8 -sourceJobHandle=3 -jobEntryId=1 -userId=kqlee;
Command OK (stuTodQueryReadInformationEfqEntryPut)
```

TodSendEfxCreate

syntax :

TodSendEfxCreate ***-todHandle=<int>*** *-dodHandle=<int>$_{optt}$ -modeOfDataTransfer=<enum>$_{opt}$ -dataSource={<ipaddress$_{slmIp}$><int$_{fuHandle}$>}$_{opt}$ -dataHandle=<int>$_{opt}$ -notificationMode={(<enumlist$_{jobStatusCode}$>) <bool$_{entries}$>}$_{opt}$ -notificationScheme={<ipaddress$_{slmIp}$><int$_{fuHandle}$>}$_{opt}$ -retryCount=<int>$_{opt}$ -retryInterval=<int>$_{opt}$ -queryInterval=<int>$_{opt}$ -coverSheetGeneration=<bool>$_{opt}$ -pageHeaderGeneration=<bool>$_{opt}$ -priority=<int>$_{opt}$*

description :

Create an extended fax TOD for use in 'EfxSendEfx' or 'RmtSendEfx' commands. This is the first in a series of nested declarations which are necessary to build an extended fax TOD. After issuing this command, the user may set the SenderUserInfo (TodSendEfxSuiSet) and the FaxSendInfo(TodSendEfxFsiSet) fields. The user must declare one or more SendDestinationInfo (TodSendEfxSdiPut) which result in fax jobs being created on the remote (or local) server when the TOD is queued remotely or locally.

parameters:

Parameter	Description
-todHandle	FU unique handle assigned to TOD by this command
-dodHandle	FU unique handle assigned to DOD when it is created. Including a dodHandle will create an additional reference to that DOD.
-modeOfDataTransfer	Default modeOfDataTransfer; Enumerated list – immediate == 0, delayed == 1
-dataSource	Default dataSource from which data is to be transferred
-dataHandle	Default handle of data to be transferred for document
-notificationMode	Default notification mode. Parameter includes a set of Job Status Codes and a boolean value allowing for a request for Job Entry Status. Only a Client FU can receive job status notifications. Enumerated list – see Salutation Architecture Specification, V2.0, Part 2, Page 201 for a listing of job status codes.
-notificationScheme	Default notification scheme. This parameter allows for notification to an FU other than the one that has initiated the job. Only a Client FU can receive job status notifications.
-retryCount	Default count value for FU retries to call the remote telephone (when the remote telephone was busy)
-retryInterval	Default interval for FU retries to call the remote telephone (expressed in minutes)
-queryInterval	Default value for when the client application needs to call to get Read Information. [FaxData] FU calls at “queryInterval” minute intervals.
-coverSheetGeneration	Default value for whether a cover sheet will be generated. Boolean – include cover sheet == True, no cover sheet == False
-pageHeaderGeneration	Default value for whether a page header will be added to the image. Boolean – include header == True, no header== False
-priority	Default priority. Integer (0..100) -- 50 is the normal priority, -- 100 is the highest priority. Valid values are enumerated in the capability attributes on a per FU basis.

result :

Command OK

notes :

TodHandles above 0x1000 (4096) are reserved for system use. Do not assign values above this value. This command performs no parameter range and bounds checking.
The DataSource-ipAddress and NotificationScheme-ipAddress must be resolvable into an actual SLMID at the time this TOD is created. The command will search for the ID based on the supplied IP address - if it is not found, then the command will return an error.

example :

```
> TodSendEfxCreate -todHandle=4 -dodHandle=1 -priority=100;
Command OK (stuTodSendEfxCreate)
```

TodSendEfxSuiSet

syntax :

TodSendEfxSuiSet ***-todHandle=<int>*** *-senderUserID=<string>$_{opt}$ -sourceFaxNumber=<string>$_{opt}$ -name=<string>$_{opt}$ -section=<string>$_{opt}$ -company=<string>$_{opt}$ -phoneNumber=<string>$_{opt}$ -address=<string>$_{opt}$*

description :

Add a sendUserInfo (SUI) declaration to a SendEfx TOD. SendUserInfo declaration is optional, and if it is not made (if this command is not part of the TOD definition sequence), no SUI will be transmitted when the job is queued.

parameters:

Parameter	Description
-todHandle	TOD handle assigned by user in TodSendEfxCreate
-senderUserID	These parameters are specified in Salutation Architecture Specification, part C
-sourceFaxNumber	
-name	
-section	
-company	
-phoneNumber	
-address	

result :

Command OK
Command ERROR

example :

```
> TodSendEfxSuiSet -todHandle=4 -senderUserId=kqlee -sourceFaxNumber="303-687-1234" -name=kerry;
Command OK (stuTodSendEfxSuiSet)
```

TodSendEfxFsiSet

syntax :

TodSendEfxFsiSet ***-todHandle=<int>*** *-issueTime=<string>$_{opt}$* *-comment=<string>$_{opt}$*

description :

Set the FaxSendInfo values for the declared TOD. This is an optional aggregate field, and if it is not set, then no FaxSendInfo structure will be used in transactions with this TOD.

parameters:

Parameter	Description
-todHandle	TOD handle assigned by user in TodSendEfxCreate
-issueTime	String listing date of issue of fax
-comment	String with comments

result :

Command OK
Command ERROR

example :

```
> TodSendEfxFsiSet –todHandle=4 -issueTime="11 August" -comment="example comment";
Command OK (stuTodSendEfxFsiSet)
```

TodSendEfxSdiPut

syntax :

TodSendEfxSdiPut *-todHandle=<int> -sdiHandle=<int> -destinationFaxNumber=<string>*
-orderingData=<string>$_{opt}$ -scheduledDateTime=<string>$_{opt}$ -scheduledAfterTime=<string>$_{opt}$

description :

Put an Efx (Extended Fax) SendDestinationInfo to a declared TOD. This will result in the definition of a job on the remote FU which is given this TOD. The sdiHandle is assigned by the user and must be referenced in subsequent commands which declare additional properties of this field. The user may declare a ReadConfirmationOrder for this SendDestinationInfo (TodSendEfxSdiRcoiSet), and the user must declare one or more entries to this field (TodSendEfxSdiEntryPut).

parameters:

Parameter	Description
-todHandle	TOD handle assigned by user in TodSendEfxCreate
-sdiHandle	User assigned SendDestinationInfo handle
-destinationFaxNumber	Telephone number string
-orderingData	Telephone number string
-scheduledDateTime	Time for scheduled future delivery. Expressed in UTCTime format.
-scheduledAfterTime	Time delay before delivery. Expressed in "hh:mm:ss" format.

result :

Command OK
Command ERROR

example :

```
> TodSendEfxSdiPut -todHandle=4 -sdiHandle=1 -destinationFaxNumber="303-987-2354";
Command OK (stuTodSendEfxSdiPut)
```

TodSendEfxSdiRcoiSet

syntax :

TodSendEfxSdiRcoiSet *-todHandle=<int> -sdiHandle=<int> -returnMethod=<enum> -timeOut=<int> -callDirectionOrder=<enum>*

description :

Define a ReadConfirmationOrder structure associated with a particular SendDestinationInfo. This is an optional structure, and if not defined, it will not be transmitted by commands using this TOD.

parameters:

Parameter	Description
-todHandle	TOD handle assigned by user in TodSendEfxCreate
-sdiHandle	Send destination info handle assigned by user in TodSendEfxSdiPut
-returnMethod	Enumerated Value – individualReturn == (0), Individual Return is ordered groupReturn == (1) Group Return is ordered
-timeOut	Read information will come in timeOut minutes If there are users who do not read this fax message after timeOut minutes, unread information is delivered
-callDirectionOrder	Enumerated Value – sourceCallsDestination == (0), -- The caller calls for Read Information. callSourceOrSourceCalls == (1), -- Call the caller for Read Information. If the callee rejects this request, the caller calls. callSourceOrGiveUp == (2) -- Call the caller for Read Information. If the callee rejects this request, the caller gives up Read Confirmation.

result :

Command OK
Command ERROR

example :

```
> TodSendEfxSdiRcoiSet -todHandle=4 -sdiHandle=1 -returnMethod=0 -timeOut=5 -
callDirectionOrder=2;
Command OK (stuTodSendEfxSdiRcoiSet)
```

TodSendEfxSdiEntryPut

syntax :

TodSendEfxSdiEntryPut ***-todHandle=<int>*** ***-sdiHandle=<int>*** ***-jobEntryId=<int>*** ***-receiverUserID=<string>*** *-readConfirmationOrder=<enum>$_{opt}$* *-name=<string>$_{opt}$* *-section=<string>$_{opt}$* *-company=<string>$_{opt}$* *-phoneNumber=<string>$_{opt}$* *-address=<string>$_{opt}$*

description :

Add a job entry to an Efx job transfer definition. Each SDI (job) must have one or more of these entries defined.

parameters:

Parameter	Description
-todHandle	TOD handle assigned by user in TodSendEfxCreate
-sdiHandle	TOD unique SDI handle assigned by user in TodSendEfxSdiPut
-jobEntryId	SDI Unique Id assigned to Job Entry
-receiverUserID	User ID of the receiver
-readConfirmationOrder	orderReadConfirmation == (0) notOrderReadConfirmation == (1)
-name	String value for fax name field
-section	String value for fax section field
-company	String value for fax company field
-phoneNumber	TelephoneNumberString value for telephone phone number field
-address	String value for fax address field

result :

Command OK
Command ERROR

example :

```
> TodSendEfxSdiEntryPut -todHandle=4 -sdiHandle=1 -jobEntryId=1 -receiverUserId=kqlee ;
Command OK (stuTodSendEfxSdiEntryPut)
```

TodDodSet

syntax :

TodDodSet ***-todHandle=<int> -dodHandle=<int>***

description :

Assign a dodHandle to the specified transfer object definition.

parameters:

Parameter	Description
-todHandle	FU unique handle assigned to TOD when it was created
-dodHandle	FU unique handle assigned to DOD when it was created. Including a dodHandle will create an additional reference to that DOD.

result :

Command OK
Command ERROR

notes :

This command must use a valid dodHandle, since it will increment the reference count of the specified DOD.

example :

```
> todDodSet -todHandle=1 -dodHandle=1;
Command OK (stuTodDodSet)
```

TodListShow

syntax :

TodListShow

description :

Print the TOD handles of each transfer object definition declared on this FU.

result :

Command OK

example :

```
> TodListShow;
TOD_LIST (handles) : 1 2 3 4
Command OK (stuTodListShow)
```

TodShow

syntax :

TodShow *-todHandle*=<*int*>

description :

Print the details of the specified transfer object definition to the output window.

parameters:

Parameter	Description
-todHandle	FU unique handle assigned to TOD when it was created

result :

Command OK
Command ERROR

example :

```
> todShow -todHandle=3;
TOD(3) type SendFax (SFX)
notificationMode-jobStatusCode( 1 2 3 4  )
notificationMode-requestJobEntriesStatus=false
dodHandle=1
Command OK (stuTodShow)
```

Data Object Definition (Dod)

The underlying SDK (upon which the test tool is based) has a common definition for all data objects - whether they are Print, Fax, or Document data objects. In order to transfer data on the salutation interface, a data object must be created from some source. The test tool supports two ways of creating such data - a simple text pattern (DodCreateFromChars) or by reading in a file. Both methods create a file which is declared as Plain-Text type data. Data may also be brought into the local environment by performing a 'requestDataTransfer' command, or by the acceptance of queued jobs or stored documents (if the tool is a server FU).

Data objects have a unique handle and are persistent (will remain until removed either manually or by an SDK level operation). A data object created from the command line will exist until it is explicitly removed. A data object belonging to a queued job or stored document will be removed when that job or document is removed. Data objects may be referenced multiple times - such as declared in multiple TOD definitions - and the data object will remain until all references to it are removed.

Most transfer object definitions will have an associated DOD. The DOD is the actual data which will be transferred by the salutation protocol, and the remainder of the TOD is parameterization and attributes of the operation to be performed on the data.

The following are the available Data Object Definition commands:

- DodCreateFromChars
- DodCreateFromFile
- DodDestroy
- DodListShow
- DodShow

DodCreateFromChars

syntax :

DodCreateFromChars ***-dodHandle=<int> -chr=<char> -chcount=<int> -segcount=<int> -blockcount=<int>***

description :

Create a salutation data object of a specified size and assign it to the specified data object definition handle (-dodHandle). The entire data object will be filled with a single repeated character (-chr).

parameters:

Parameter	Description
-dodHandle	A unique data handle that will be used to reference the data object. Data handles above 0x1000 (4096) are reserved for system use. Assigning a data handle above this value may cause unpredictable system behavior.
-chr	Fill character for reserved data blocks (ASCII character number)
-chcount	Number of characters in a segment - limited in size to 0x8000
-segcount	Number of segments in a block
-blockcount	Number of blocks in a data object

result :

Command OK
Command ERROR

notes :

The size of all created data is limited and defined when the Fu is created. This is limit on this individual data object (chcount x segcount x blockcount) and on the total of all user and system created data objects.

example :

```
> DodCreateFromChars -dodHandle=3 -chr=48 -chcount=10000 -segcount=2 -blockcount=2;
Command OK (stuDodCreateFromChars)
> DodListShow;
DOD_LIST (handles) : 1 3
Command OK (stuDodListShow)
> DodShow -dodHandle=3;
DODNODE (3 ) (dataHandle=3,size=40000,blkcnt=2,bufcnt=10,refCnt=1)
Command OK (stuDodShow)
```

DodCreateFromFile

syntax :

DodCreateFromFile *-dodHandle=<int> -filename=<string>*

description :

Create a salutation data object from the specified file. The data object will have a single block, will be of format "PLAIN-TEXT", and will have as many segments as required to hold the file's data.

parameters:

Parameter	Description
-dodHandle	A unique data handle that will be used to reference the data object. Data handles above 0x1000 (4096) are reserved for system use. Assigning a data handle above this value may cause unpredictable system behavior.
-filename	File name from which the data object was created. A DOS path can be included as a part of the file name string. The default DOS directory is the directory from where FuMaster.exe was launched.

result :

Command OK
Command ERROR

example :

```
> DodCreateFromFile -dodHandle=4 -filename="startup";
Command OK (stuDodCreateFromFile)
> DodListShow;
DOD_LIST (handles) : 1 3 4
Command OK (stuDodListShow)
> DodShow -dodHandle=4;
DODNODE (4 ) (dataHandle=4,size=5024,blkcnt=1,bufcnt=4,refCnt=1)
Command OK (stuDodShow)
```

DodDestroy

syntax :

DodDestroy ***-dodHandle=<int>***

description :

Destroy a user or system created data object. This command will remove the implicit reference which is made to a Data Object Definition when it is created. If additional references have been made to the DOD since it was created, then the DOD will not actually be destroyed until those references are also deleted.

parameters:

Parameter	Description
-dodHandle	Data handle of the data object to be destroyed

result :

Command OK
Command ERROR

notes :

This command simply decrements a counter that tracks the number of references made to the DOD. If you delete the last DOD reference, then the DOD is destroyed. Any commands that reference the DOD will then error.

example :

```
> DodDestroy -dodHandle=3;
Command OK (stuDodDestroy)
```

DodListShow

syntax :

DodListShow

description :

List the local data object handles and spool area information to the console window.

result :

Command OK - List (possibly empty) of handles to data objects stored on FU.

example :

```
DOD_LIST (handles) : 23 22
DOD Spool Area : Total = 1048576 , Remaining = 532064
Command OK (stuDodListShow)
```

DodShow

syntax :

DodShow *-dodHandle=<int>*

description :

List the details of the specified local data object to the console window.

parameters:

Parameter	Description
-dodHandle	Data handle of the data object details to be listed

result :

Command OK - details of data object definition (DOD)
Command Error

example :

```
> DodShow -dodHandle=4;
DODNODE (4 ) (dataHandle=4,size=5024,blkcnt=1,bufcnt=4,refCnt=1)
Command OK (stuDodShow)
```

Global Attribute Commands

The global attribute commands are relevant to local attributes only. To query global attributes on a remote FU, you must use the appropriate 'Rmt' command. Since the salutation protocol does not support remote setting of global attributes, they may only be set with these local commands.

The test tool automatically registers a full set of capability attributes for each defined FU type.

The following are the available Global Attribute commands:

- GlobalAttributeSet
- GlobalAttributeGet
- GlobalAttributeListShow

GlobalAttributeSet

syntax :

GlobalAttributeSetInt *-attributeId=<int> -attributeValue=<int>*

GlobalAttributeSetEnum *-attributeId=<int> -attributeValue=<enum>*

GlobalAttributeSetBool *-attributeId=<int> -attributeValue=<bool>*

GlobalAttributeSetString *-attributeId=<int> -attributeValue=<string>*

description :

Set a global attribute value. There is no analog to this command on the salutation interface. Global attributes may only be set locally - they can not be set through the salutation protocol. Attributes must have been registered as Capability attributes before they can be set.

parameters:

Parameter	Description
-attributeId	Local FU attribute Id to be set. See Salutation Architecture Specification, V2.0, Part 2, section 9.0 for a listing of attribute Id's
-attributeValue	Value to which attribute will be set

result :

Command OK
Command ERROR

notes :

1) Range and bounds checking are disabled for local attribute commands. Illegal values may be set in valid global attributeId's.
2) The SDK allows non-standard (vendor) ID's to be freely set - though no type or bounds checking is performed on these values. They may be queried through the salutation protocol (GlobalAttributeGet). Caution should be exercised in using this facility since it is not architecturally defined.
3) This command will not work for aggregate values. It will only work with the four listed primitive value types.

example :

```
> GlobalAttributeSetInt -attributeId=10040 -attributeValue=50;
Command OK (stuGatSetInt)
> GlobalAttributeGet -attributeId=10040;
int-value = 50
Command OK (stuGatGet)
```

GlobalAttributeGet

syntax :

GlobalAttributeGet *-attributeId=<int>*

description :

Show a global attribute value.

parameters:

Parameter	Description
-attributeId	Local FU attribute Id to be shown. See Salutation Architecture Specification, V2.0, Part 2, section 9.0 for a listing of attribute Id's

result :

Command OK - display attribute value
Command ERROR

example :

```
> GlobalAttributeSetInt -attributeId=10040 -attributeValue=50;
Command OK (stuGatSetInt)
> GlobalAttributeGet -attributeId=10040;
int-value = 50
Command OK (stuGatGet)
```

GlobalAttributeListShow

syntax :

GlobalAttributeListShow

description :

Show a list of global attribute id's on local FU.

result :

Command OK - list (possibly empty) of attribute ID's

example :

```
> GlobalAttributeListShow;
GATT_LIST (handles) : 10020 10021 10022 10024 10025 10026 10027 10028 10029 10030 10041 10044 445 10040
Command OK (stuGatListShow)
```

Private Attribute Commands

The private attribute commands are relevant to local attributes only. To set or get private attributes on a remote FU, you must use the appropriate 'Rmt' command. Private attributes are specific to a transaction on a particular session, and override any default or global attribute values.

The following are the available Private Attribute commands:

PrivateAttributeSet
PrivateAttributeGet
PrivateAttributeListShow

PrivateAttributeSet

syntax :

PrivateAttributeSetInt ***-sessionHandle=<int>*** ***-attributeId=<int>*** *-attributeValue=<int>*$_{opt}$

PrivateAttributeSetEnum ***-sessionHandle=<int>*** ***-attributeId=<int>*** *-attributeValue=<enum>*$_{opt}$

PrivateAttributeSetBool ***-sessionHandle=<int>*** ***-attributeId=<int>*** *-attributeValue=<bool>*$_{opt}$

PrivateAttributeSetString ***-sessionHandle=<int>*** ***-attributeId=<int>*** *-attributeValue=<string>*$_{opt}$

description :

Set the specified private attribute to the indicated value. If no value is indicated, the attribute value should be removed from the session's private attribute list.

parameters:

Parameter	Description
-sessionHandle	FU unique handle assigned to session when it is opened on local FU
-attributeId	Local FU attribute Id to be set. See Salutation Architecture Specification, V2.0, Part 2, section 9.0 for a listing of attribute Id's
-attributeValue	Value to which attribute will be set

result :

Command OK
Command ERROR

notes :

1) Range and bounds checking are disabled for local attribute commands. Illegal values may be set in valid private attributeId's.
2) The SDK allows non-standard (vendor) ID's to be freely set - though no type or bounds checking is performed on these values. They may be queried through the salutation protocol (PrivateAttributeGet). Caution should be exercised in using this facility.
3) This command will not work for aggregate values. It will only work with the four listed primitive value types.

example :

```
> PrivateAttributeSetEnum -sessionHandle=5311728 -attributeId=10040 -attributeValue=50;
Command OK (stuPatSetEnum)
```

PrivateAttributeGet

syntax :

PrivateAttributeGet *-sessionHandle=<int> -attributeId=<int>*

description :

Show a private attribute value.

parameters :

Parameter	Description
-sessionHandle	FU unique handle assigned to session when it is opened on local FU
-attributeId	Local FU attribute Id to be shown. See Salutation Architecture Specification, V2.0, Part 2, section 9.0 for a listing of attribute Id's

result :

Command OK - decoded value of private attribute
Command ERROR

example :

```
> PrivateAttributeGet -sessionHandle=5311728 -attributeId=10040;
enum-value = 50
Command OK (stuPatGet)
```

PrivateAttributeListShow

syntax :

PrivateAttributeListShow *-sessionHandle=<int>*

description :

Show private attribute list (id's of private/session attributes on a particular session on the local FU).

parameters:

Parameter	Description
-sessionHandle	FU unique handle assigned to session when it is opened on local FU

result :

Command OK - display a list (possibly empty) of attributes which are set on the specified session.

example :

```
> PrivateAttributeListShow -sessionHandle=5311728;
PATT_LIST (handles) : 10040
Command OK (stuDodListShow)
```

Job Commands

These commands control the job queue and its jobs that are maintained on the local FU. To perform job commands on a remote FU, use the appropriate 'Rmt' commands. Jobs can be placed on the job queue by either local or remote FU's. Regardless of how the job gets on the queue, it is treated the same once it is queued.

The following are the available Job commands:

- JobQEnable
- JobQDisable
- JobQClear
- JobStart
- JobComplete
- JobCancel
- JobAbort
- JobError
- JobAttributeGet
- JobAttributeSet
- JobListShow
- JobShow

JobQEnable

syntax :

JobQEnable

description :

Allow jobs to be queued by a remote FU. If the jobQ is not enabled, remote job commands (RmtPrint, RmtSendSfx, RmtSendEfx) will be rejected by the local FU.

result :

Command OK

example :

```
> jobQEnable;
Command OK (stuJobQEnable)
```

JobQDisable

syntax :

JobQDisable

description :

Do not allow jobs to be queued to the local FU by remote commands. Remote commands which attempt to queue jobs will be rejected if the queue is disabled.

result :

Command OK

example :

```
> jobQDisable;
Command OK (stuJobQDisable)
```

JobQClear

syntax :

JobQClear

description :

Command will delete any jobs and job entries in the job queue.

result :

Command OK

notes :

It is strongly recommended that this operation only be performed while the jobQ is disabled. Jobs deleted by this command are completely, immediately, and extra-architecturally erased. No notifications are made to remote clients.

example :

```
> JobQDisable;
Command OK (stuJobQDisable)
> JobQClear;
Command OK (stuJobQClear)
```

JobStart

syntax :

JobStart *-jobHandle=<int>*

description :

Set the job status to the specified job to “started”. Notify any job status subscribers of this change in status.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.

result :

Command OK
Command ERROR

notes :

A job may be set to this state regardless of its former state. This is under control of the user - though the architecture does define a progression (sequence) of valid states, this is not enforced by this tool.

example :

```
> jobStart -jobHandle=4096;
Command OK (stuJobStart)
```

JobComplete

syntax :

JobComplete *-jobHandle=<int>*

description :

Set the job status of the specified job to complete. Notify any job status subscribers of this change in status. If the job's life is 'JOB' (0), then remove the job from the job queue and free its associated resources.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.

result :

Command OK
Command ERROR

notes :

A job may be set to this state regardless of its former state. This is under control of the user - though the architecture does define a progression (sequence) of valid states, this is not enforced by this tool.

example :

```
> jobComplete -jobHandle=4096;
Command OK (stuJobComplete)
```

JobCancel

syntax :

JobCancel *-jobHandle=<int>*

description :

Set the job status of the specified job to cancelled. Notify any job status subscribers of this change in status. If the job's life is 'JOB' (0), then remove the job from the job queue and free its associated resources.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.

result :

Command OK
Command ERROR

notes :

Canceling a job should be done before the job is started. If a job is already started, then use the JobAbort command.
A job may be set to this state regardless of its former state. This is under control of the user - though the architecture does define a progression (sequence) of valid states, this is not enforced by this tool.

example :

```
> jobCancel -jobHandle=4097;
Command OK (stuJobCancel)
```

JobAbort

syntax :

JobAbort *-jobHandle=<int>*

description :

Set the job status of the specified job to aborted. Notify any job status subscribers of this change in status. If the job's life is 'JOB', then remove the job from the job queue and free its associated resources.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.

result :

Command OK
Command ERROR

notes :

Aborting a job should be done after the job is started. If a job is queued and not yet started, then use the JobCancel command.
A job may be set to this state regardless of its former state. This is under control of the user - though the architecture does define a progression (sequence) of valid states, this is not enforced by this tool.

example :

```
> jobAbort -jobHandle=4098;
Command OK (stuJobAbort)
```

JobError

syntax :

JobError *-jobHandle=<int>*

description :

Set the job status of the specified job to the error state. Notify any job status subscribers of this change in status. If the job's life is 'JOB', then remove the job from the job queue and free its associated resources

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.

result :

Command OK
Command ERROR

notes :

A job may be set to this state regardless of its former state. This is under control of the user - though the architecture does define a progression (sequence) of valid states, this is not enforced by this tool.

example :

```
> JobError -jobHandle=4099;
Command OK (stuJobError)
```

JobAttributeGet

syntax :

JobAttributeGet *-jobHandle=<int> -attributeId=<int>*

description :

Get a job attribute value. There is no remote analogue to this command, so this is the only way to see what values have been set on a queued job.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.
-attributeId	Id of the job attribute.

result :

Command OK
Command ERROR

notes :

Job attributes must have been first declared and registered before operations on them will be allowed. The test tool automatically registers a full set of capability attributes for each defined FU type, so this should not pose a problem for users.

example :

```
> JobAttributeGet -jobHandle=4096 -attributeId=10040;
enum-value = 50
Command OK (stuJobAttributeGet)
```

JobAttributeSet

syntax :

JobAttributeSetInt *-jobHandle=<int> -attributeId=<int> -attributeValue=<int>*

JobAttributeSetEnum *-jobHandle=<int> -attributeId=<int> -attributeValue=<enum>*

JobAttributeSetBool *-jobHandle=<int> -attributeId=<int> -attributeValue=<bool>*

JobAttributeSetString *-jobHandle=<int> -attributeId=<int> -attributeValue=<string>*

description :

Set a job attribute value.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.
-attributeId	Id of the job attribute
-attributeValue	Value of the attribute

result :

Command OK
Command ERROR

notes :

1) Job attributes must have been first declared and registered before operations on them will be allowed. The test tool automatically registers a full set of capability attributes for each defined FU type, so this should not pose a problem for users.
2) Range and bounds checking are disabled for local attribute commands. Illegal values may be set in valid job attributeId's.
3) The SDK allows non-standard (vendor) ID's to be freely set - though no type or bounds checking is performed on these values. They may be queried through the salutation protocol (JobAttributeGet). Caution should be exercised in using this facility.
4) This command will not work for aggregate values. It will only work with the four listed primitive value types.

example :

```
> JobAttributeSetEnum -jobHandle=4096 -attributeId=10040 -attributeValue=100;
Command OK (stuJobAttributeSetEnum)
```

JobListShow

syntax :

JobListShow

description :

Display a list of all job handles on the job queue.

result :

Command OK - list (possibly empty) of all job handles queued on the local FU.

notes :

Each FU which uses jobs will have a more FU specific way of listing jobs (such as PrnListPrintJob). This is just a generic jobQ utility and will return much less detail.

example :

```
> JobListShow;
JOB_LIST (handles) : 4100
Command OK (stuJobListShow)
```

JobShow

syntax :

JobShow *-jobHandle=<int>*

description :

Display details of the specified job.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.

result :

Command OK
Command ERROR

notes :

Each FU which uses jobs will have a more FU specific way of listing jobs (such as PrnListPrintJob). This is just a generic jobQ utility and will return different detail.

example :

```
> JobShow -jobHandle=4100;
Job[4100] :
hfu               = 1
slmid              = { slmId=0:0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0 }
hsvc               = 0
jobStatusCode      = 1
jobPrevStatusCode  = 0
jobType            = 10000
queuedTime         =
requesterUserId    =
dataSource         = { slmId=5:c0.9.c8.c9.f9.6c.ed.34.0.0.0.0.0.0.0 fuHandle=1 }
dataHandle         = 1
notificationMode   = { ( 1 2 3 4) , false}
notificationScheme = { slmId=5:c0.9.c8.c9.f9.6c.ed.34.0.0.0.0.0.0.0 fuHandle=1 }
JATT_LIST (handles) : 10044 10041 996 10020 10021 10022 10023 10024 10025 10026 10027 10028 10029 10040 10030
10999 997
dodHandle          = 1
Command OK (stuJobShow)
```

Job Entry Command

Job Entry Commands control job entries that are maintained on the local FU. To perform job entry commands on a remote FU, use the appropriate 'Rmt' commands. Changing the state of job entries has no effect on the state of the parent job. Job status must be controlled manually and separately from job entry status.

The following are the available Job Entry commands:

- JobEntryStart
- JobEntryComplete
- JobEntryCancel
- JobEntryAbort
- JobEntryError
- JobEntryAttributeSet
- JobEntryAttributeGet
- JobEntryListShow
- JobEntryShow

JobEntryStart

syntax :

JobEntryStart *-jobHandle=<int> -jobEntryId=<int>*

description :

Set the status of the specified jobentry to start. Notify any job status subscribers of this change in status.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.
-jobEntryId	Unique Id assigned by client to Job Entry

result :

Command OK
Command ERROR

notes :

A job may be set to this state regardless of its former state. This is under control of the user - though the architecture does define a progression (sequence) of valid states, this is not enforced by this tool.

example :

```
> JobEntryStart -jobHandle=4096 -jobEntryId=1;
Command OK (stuJobEntryStart)
```

JobEntryComplete

syntax :

JobEntryComplete *-jobHandle=<int> -jobEntryId=<int>*

description :

Set the status of the specified jobentry to complete. Notify any job status subscribers of this change in status.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.
-jobEntryId	Unique Id assigned by client to Job Entry

result :

Command OK
Command ERROR

notes :

A job may be set to this state regardless of its former state. This is under control of the user - though the architecture does define a progression (sequence) of valid states, this is not enforced by this tool.

example :

```
> JobEntryComplete -jobHandle=4096 -jobEntryId=1;
Command OK (stuJobEntryComplete)
```

JobEntryCancel

syntax :

JobEntryCancel *-jobHandle=<int> -jobEntryId<int>*

description :

Set the status of the specified jobentry to cancelled. Notify any job status subscribers of this change in status.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.
-jobEntryId	Unique Id assigned by client to Job Entry

result :

Command OK
Command ERROR

notes :

A job may be set to this state regardless of its former state. This is under control of the user - though the architecture does define a progression (sequence) of valid states, this is not enforced by this tool.

example :

```
> JobEntryCancel -jobHandle=4096 -jobEntryId=1;
Command OK (stuJobEntryCancel)
```

JobEntryAbort

syntax :

JobEntryAbort *-jobHandle=<int> -jobEntryId=<int>*

description :

Set the status of the specified jobentry to aborted. Notify any job status subscribers of this change in status.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.
-jobEntryId	Unique Id assigned by client to Job Entry

result :

Command OK
Command ERROR

notes :

A job may be set to this state regardless of its former state. This is under control of the user - though the architecture does define a progression (sequence) of valid states, this is not enforced by this tool.

example :

```
> JobEntryAbort -jobHandle=4096 -jobEntryId=1;
Command OK (stuJobEntryAbort)
```

JobEntryError

syntax :

JobEntryError *-jobHandle=<int> -jobEntryId=<int>*

description :

Set the status of the specified jobentry to errored. Notify any job status subscribers of this change in status.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.
-jobEntryId	Unique Id assigned by client to Job Entry

result :

Command OK
Command ERROR

notes :

A job may be set to this state regardless of its former state. This is under control of the user - though the architecture does define a progression (sequence) of valid states, this is not enforced by this tool.

example :

```
> JobEntryError -jobHandle=4096 -jobEntryId=1;
Command OK (stuJobEntryError)
```

JobEntryAttributeSet

syntax :

JobEntryAttributeSetInt *-jobHandle=<int> -jobEntryId=<int> -attributeId=<int> -attributeValue=<int>*

JobEntryAttributeSetEnum *-jobHandle=<int> -jobEntryId=<int> -attributeId=<int> -attributeValue=<enum>*

JobEntryAttributeSetBool *-jobHandle=<int> -jobEntryId=<int> -attributeId=<int> -attributeValue=<bool>*

JobEntryAttributeSetString *-jobHandle=<int> -jobEntryId=<int> -attributeId=<int> -attributeValue=<string>*

description :

Set a job entry attribute value.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.
-jobEntryId	Unique Id assigned by client to Job Entry
-attributeId	Id of the job attribute.
-attributeValue	Value of the attribute

result :

Command OK
Command ERROR

notes :

1) Job attributes must have been first declared and registered before operations on them will be allowed. The test tool automatically registers a full set of capability attributes for each defined FU type, so this should not pose a problem for users.
2) Range and bounds checking are disabled for local attribute commands. Illegal values may be set in valid job attributeId's.
3) The SDK allows non-standard (vendor) ID's to be freely set - though no type or bounds checking is performed on these values. They may be queried through the salutation protocol (JobAttributeGet). Caution should be exercised in using this facility.
4) This command will not work for aggregate values. It will only work with the four listed primitive value types.

example :

```
> JobEntryAttributeSetEnum -jobHandle=4096 -jobEntryId=1 -attributeId=12031 -attributeValue=50;
Command OK (stuJobEntryAttributeSetEnum)
```

JobEntryAttributeGet

syntax :

JobEntryAttributeGet *-jobHandle=<int> -jobEntryId=<int> -attributeId=<int>*

description :

Show a job entry attribute value.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.
-jobEntryId	Unique Id assigned by client to Job Entry
-attributeId	Id of the job attribute.

result :

Command OK
Command ERROR

notes :

Job entry attributes must have been first declared and registered before operations on them will be allowed. The test tool automatically registers a full set of capability attributes for each defined FU type, so this should not pose a problem for users.

example :

```
> JobEntryAttributeGet -jobHandle=4096 -jobEntryId=1 -attributeId=12031;
enum-value = 50
Command OK (stuJobEntryAttributeGet)
```

JobEntryListShow

syntax :

JobEntryListShow *-jobHandle=<int>*

description :

Show a list of all job entries associated with a particular job.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.

result :

Command OK - list (possibly empty) of job entries on specified job

example :

```
> JobEntryListShow -jobHandle=4096;
JOB_ENTRY_LIST (handles) : 1
Command OK (stuJobEntryQShow)
```

JobEntryShow

syntax :

JobEntryShow *-jobHandle=<int> -jobEntryId=<int>*

description :

Show details of a particular job entry.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.
-jobEntryId	Unique Id assigned by client to Job Entry

result :

Command OK - details of job entry
Command ERROR

example :

```
> jobEntryShow -jobHandle=4096 -jobEntryId=1;
JOBENTRYNODE(1 ) Job Entry [1] :
jobEntrystatusCode = 0
hfu                = 1
jobHandle          = 4096
dodHandle          = 0
JEAT_LIST (handles) : 12899 12898 12897 12896 12895 12894 12030 12893
Command OK (stuJobEntryShow)
```

Print FU Commands (Prn)

These commands affect the status and operation of the local Print FU. To perform print commands on a remote FU, use the appropriate 'Rmt' commands.

The following are the available local print FU commands:

PrnDsOperationStatus
PrnDsErrorDetail
PrnDsPaperInputTrayStatus
PrnListPrintJob
PrnPrint

The following Dynamic Status Parameters are defined for [Print] Functional Unit.

Dynamic Status Parameter	Query	Event	ID	Description
PrinterOperationStatus	Yes	Yes	10000	status of printing equipment.
PrinterErrorDetail	Yes	No	10001	detail error information of equipment's.
FreeStorageSize	Yes	No	10002	available storage size, K Byte.
PrinterPaperInputTray	Yes	No	10003	status of paper size and direction in each input tray.
ListExcerptPrintJob	Yes	Yes	10004	lists a excerpt from print job descriptions

PrnDsOperationStatus

syntax :

PrnDsOperationStatus *-noPaper=<bool>$_{opt}$ -noToner=<bool>$_{opt}$ -doorOpen=<bool>$_{opt}$ -jammed=<bool>$_{opt}$ -offline=<bool>$_{opt}$ -receiving=<bool>$_{opt}$ -error=<bool>$_{opt}$ -normal=<bool>$_{opt}$ -paperNearEnd=<bool>$_{opt}$ -tonerNearEnd=<bool>$_{opt}$ -fatalError=<bool>$_{opt}$*

description :

Change the operation status (dynamic status ID 10000) of the print FU. Set the specified status parameters only. Notify any clients subscribed to this event of the change in status (reporting all true status fields).

parameters:

Parameter	Description	
-noPaper=<bool>$_{opt}$	noPaper	(0),
-noToner=<bool>$_{opt}$	noToner	(1),
-doorOpen=<bool>$_{opt}$	doorOpen	(2),
-jammed=<bool>$_{opt}$	jammed	(3),
-offline=<bool>$_{opt}$	offline	(4),
-receiving=<bool>$_{opt}$	receiving	(5),
-error=<bool>$_{opt}$	error	(6),
-normal=<bool>$_{opt}$	normal	(7),
-paperNearEnd=<bool>$_{opt}$	paperNearEnd	(8),
-tonerNearEnd=<bool>$_{opt}$	tonerNearEnd	(9),
-fatalError=<bool>$_{opt}$	fatalError	(10),
	others	(127)

result :

Command OK

notes :

Only the parameters included on the command line will be changed. All other parameters will remain as previously set or as set by default. This command should be issued only to a Print FU. Client FU(s) should be subscribed to remote status ID 10000 if they wish to receive asynchronous event notification (notifyEvent command).

example :

```
> PrnDsOperationStatus -noPaper=true -offline=true -fatalError=false;
Command OK (stuPrnDsOperationStatus)
```

PrnDsErrorDetail

syntax :

PrnDsErrorDetail *-statusCode=<enum> -systemError=<string> -others=<string>*

description :

Set the printer error detail. Change the operation status (dynamic status ID 10001) of the print FU. Set the specified status parameters only.

parameters:

Parameter	Description
-statusCode	Enumerated list – see Salutation Architecture Specification, V2.0, Part 2, Page 80.
-systemError	String containing error description
-others	String containing other description of error

result :

Command OK

example :

```
> PrnDsErrorDetail -statusCode=2 -systemError="not working" -others="";
Command OK (stuPrnErrorDetail)
```

PrnDsPaperInputTrayStatus

*** *This command is not supported by this current release.*

syntax :

PrnDsPaperInputTrayStatus ***-inputSelect=<int>*** ***-size=<enum>*** ***-direction=<enum>*** *-existence=<bool>*$_{opt}$

description :

Set the paper input tray status. The input tray must fall within the bounds of the inputs declared when the print FU was created.

parameters:

Parameter	Description
-inputSelect	Paper tray input selection. Enumerated list - manualFeed == (0), tray-1 == (1), tray-2 == (2), tray-3 == (3), tray-4 == (4), tray-5 == (5), automaticSelect (FU selects tray by PaperSize) == (126), other == (127)
-size	Enumerated list – see Salutation Architecture Specification, V2.0, Part 2, Page 210.
-direction	Enumerated list – portrait == (1), landscape == (2), other == (127)
-existence	Lists status of paper in selected tray. Boolean – True == tray in not empty False == tray is empty

result :

Command OK

example :

```
> PrnDsPaperInputTrayStatus -inputSelect=2 -size=21 -direction=1 -existence=true;
Command OK (stuPrnPaperInputTrayStatus)
```

PrnListPrintJob

syntax :

PrnListPrintJob

description :

Show a list of print jobs on the queue. This should produce the same output as the remote list print job command would for a client.

result :

Command OK - list (possibly empty) of print jobs on queue.

example :

```
> PrnListPrintJob;
stmDecodePrintJobList
jobHandle=(4096) requesterUserId=() jobStatusCode=(1) dataSize=(400000) printPriority=(50) queuedTime=(Fri Oct 29
00:56:29 1999) printFileName=(printFile)
Command OK (stuPrnListPrintJob)
```

PrnPrint

syntax :

PrnPrint *-todHandle=<int>$_{opt}$ -dodHandle=<int>$_{opt}$ -modeOfDataTransfer=<enum>$_{opt}$ -dataSource={<ipaddress$_{slmIp}$><int$_{fuHandle}$>}$_{opt}$ -dataHandle=<int>$_{opt}$ -life=<enum>$_{opt}$ -notificationMode={(<enumlist$_{jobStatusCode}$>) <bool$_{entries}$>}$_{opt}$ -notificationScheme={<ipaddress$_{slmIp}$><int$_{fuHandle}$>}$_{opt}$ -paperSize=<enum>$_{opt}$ -resolution=<enum>$_{opt}$ -paperDirection=<enum>$_{opt}$ -copyCount=<int>$_{opt}$ -inputSelect=<enum>$_{opt}$ -outputSelect=<enum>$_{opt}$ -outputBinSelect=<int>$_{opt}$ -duplexModeSelect=<enum>$_{opt}$ -duplexBindingMargin=<int>$_{opt}$ -faceUpModeSelect=<enum>$_{opt}$ -priority=<int>$_{opt}$ -staplingSelect=<int>$_{opt}$ -fileName=<string>$_{opt}$*

description :

Create and queue a print job on the local FU. A todHandle with associated data may be specified. Any parameters specified in this command will override the defaults (if any) specified in the transfer object definition (TOD). Indicate success or indicate cause of failure.

parameters:

Parameter	Description
-todHandle	FU unique handle assigned to TOD when it is created. See caution in notes.
-dodHandle	FU unique handle assigned to DOD when it is created. Including a dodHandle will create an additional reference to that DOD. See caution in notes.
-modeOfDataTransfer	Enumerated list – immediate == 0, delayed == 1. If not defined, immediate is assumed.
-dataSource	Address of remote (3rd party) data source if data is not to be transferred from client FU. See caution in notes.
-dataHandle	Handle of remote (3rd party) data if data is not to be transferred from client FU. See caution in notes.
-life	Enumerated list – job == (0), session == (1), persistent == (2)
-notificationMode	Parameter includes a set of Job Status Codes and a boolean value allowing for a request for Job Entry Status. Only a Client FU can receive job status notifications. Enumerated list – see Salutation Architecture Specification, V2.0, Part 2, Page 201 for a listing of job status codes.
-notificationScheme	This parameter allows for notification to an FU other than the one that has initiated the job. Only a Client FU can receive job status notifications.
-paperSize	Enumerated list – see Salutation Architecture Specification, V2.0, Part 2, Page 210.
-resolution	Print resolution. Enumerated list – see Salutation Architecture Specification, V2.0, Part 2, Page 208.
-paperDirection	Paper direction. Enumerated list – portrait == (1), landscape == (2), other == (127)
-copyCount	Integer defining the number of copies requested – default is 1 copy
-inputSelect	Paper tray input selection. Enumerated list - manualFeed == (0), tray-1 == (1), tray-2 == (2), tray-3 == (3), tray-4 == (4), tray-5 == (5), automaticSelect (FU selects tray by PaperSize) == (126), other == (127)
-outputSelect	Output sortation mode. Enumerated list - standard == (0), collatedSort == (1), stack == (2), nonCollatedSort == (3), other == (127)
-outputBinSelect	Output bin number. Integer corresponding to equipment bin number
-duplexModeSelect	Duplex binding mode. Enumerated list - simplex == (0), left-binding-duplex == (1), right-binding-duplex == (2), top-binding-duplex == (3), other == (127)
-duplexBindingMargin	Margin distance for duplex binding. (0.1mm x duplexBindingMargin)??
-faceUpModeSelect	Enumerated list – faceDown == (1), faceUp == (2), other == (127)
-priority	Integer (0..100) -- 50 is the normal priority, -- 100 is the highest priority. Valid values are enumerated in the capability attributes on a per FU basis.
-staplingSelect	Stapling location. Enumerated list – see Salutation Architecture Specification, V2.0, Part 2, Page 215.
-fileName	File name of print job

result :

Command OK - new job handle
Command ERROR

notes :

CAUTION: A valid data source must be defined for this command to function properly. There are three ways that the data source can be referenced:

1 – Reference a DOD with –dodHandle parameter.
2 – Reference a TOD with –todHandle parameter. The TOD must then reference a DOD.
3 – Reference a 3rd party data source with –dataSource and –dataHandle parameters.

example :

```
> prnPrint -todHandle=1;
jobHandle=4096
Command OK (stuDocPrint)
```

DocStore FU Commands (Doc)

These commands affect the status and operation of the local DocStore FU. To perform docStore commands on a remote FU, use the appropriate 'Rmt' commands.

The following are the available local DocStore FU commands:

DocDsOperatorIntervention
DocDsOperatorInformation
DocStoreDoc
DocListFolder
DocListFolderDoc
DocMoveDoc
DocCopyDoc
DocDeleteDoc
DocChangeDocDescription
DocCreateFolder
DocDeleteFolder
DocChangeFolderDescription

The following Dynamic Status Parameters are defined for [Document Storage] Functional Unit.

Dynamic Status Parameter	Query	Event	ID	Description
FreeStorageSize	Yes	No	11000	available storage size.
OperatorIntervention	No	Yes	11001	a warning message to operator or administrator to request human intervention
OperatorInformation	No	Yes	11002	an informational message to operator or administrator

DocDsOperatorIntervention

syntax :

DocDsOperatorIntervention *-requiredAction=<string>*

description :

Set the current operator intervention string. Notify any subscribers that this dynamic status (ID=11001) has changed.

parameters:

Parameter	Description
-requiredAction	a warning message to operator or administrator to request human intervention

result :

Command OK

example :

```
> DocDsOperatorIntervention -requiredAction="reboot system";
Command OK (stuDocDsOperatorIntervention)

[Client notification]
defaultNcb()...
notifyEvent(sbsHandle=1,id=11001)
requiredAction=(reboot system)
```

DocDsOperatorInformation

syntax :

DocDsOperatorInformation *-operatorInformation=<string>*

description :

Change the currently registered operator information. Notify any subscribers that this dynamic status (ID=10002) has been updated.

parameters:

Parameter	Description
-operatorInformation	an informational message to operator or administrator

result :

Command OK

example :

```
> DocDsOperatorInformation -operatorInformation="storage is low";
Command OK (stuDocDsOperatorInformation)

[Client notification]
defaultNcb()...
notifyEvent(sbsHandle=1,id=11002)
information=(storage is low)
```

DocStoreDoc

syntax :

DocStoreDoc *-todHandle=<int>$_{opt}$ -dodHandle=<int>$_{opt}$ -folderId=<int>$_{opt}$ -dataSource={<ipaddress$_{slmIp}$><int$_{fuHandle}$>}$_{opt}$ -dataHandle=<int>$_{opt}$ -modeOfStore=<enum>$_{opt}$ -ownerName=<string>$_{opt}$ -docComment=<string>$_{opt}$*

description :

Create and store a document on the local FU. A todHandle with associated data, may be specified. Any parameters specified in this command will override the defaults (if any) previously specified in the transfer object definition.

parameters:

Parameter	Description
-todHandle	FU unique handle assigned to TOD when it is created. See caution in notes.
-dodHandle	DOD handle of data object to transfer (overrides data object id of TOD)) See caution in notes.
-folderId	ID of folder on remote FU into which the new document will go. If –todHandle parameter is not included or if the referenced TOD does not reference a DOD, then this parameter is required. See caution in notes.
-dataSource	Address of remote (3rd party) data source if data is not to be transferred from client FU. See caution in notes.
-dataHandle	Handle of remote (3rd party) data if data is not to be transferred from client FU. See caution in notes.
-modeOfStore	1 == Non-transparent Mode - transmit and store data as multiple blocks (retain block boundaries as when created) 2 == Transparent Mode - transmit and store data a single block (combine blocks) Overrides remote FU's global or private attribute
-ownerName	Name of new document owner
-docComment	Comment string for new document

result :

Command OK
Command ERROR

notes :

CAUTION: A valid Folder ID must defined for this command to function properly. The Folder ID can either be defined at the time of document creation (with this command) or by a Transfer Object Definition (TOD) which must then be referenced by a TOD Handle in this command.

CAUTION: A valid data source must be defined for this command to function properly. There are three ways that the data source can be referenced:
1 – Reference a DOD with –dodHandle parameter.
2 – Reference a TOD with –todHandle parameter. The TOD must then reference a DOD.
3 – Reference a 3rd party data source with –dataSource and –dataHandle parameters.

The modeOfStore parameter setting = 2, is not applied until data is transferred from the DocStore. While stored, the data will retain its original block count.

example :

```
> DocListFolder;
stmDecodeFolderList
folderId=(1) ownerName=(kqlee) folderComment=(example folder) createDateTime=(Tue Oct 26 01:54:01 1999) usedSize=(0)
freeSize=(0) numberOfDocuments=(0)
Command OK (stuDocListFolder)

> DocStoreDoc -dodHandle=1 -folderId=1 -ownerName=kqlee -docComment="example comment";
documentId=1
Command OK (stuDocStoreDoc)

> DocListFolder;
stmDecodeFolderList
folderId=(1) ownerName=(kqlee) folderComment=(example folder) createDateTime=(Tue Oct 26 01:54:01 1999) usedSize=(0)
freeSize=(0) numberOfDocuments=(1)
Command OK (stuDocListFolder)

> DocListFolderDoc -folderId=1;
documentId=(1) ownerName=(kqlee) docComment=(example comment) createDateTime=(Tue Oct 26 01:57:45 1999)
size=(400000) numberOfBlocks=(2)
Command OK (stuDocListFolderDoc)
```

DocListFolder

syntax :

DocListFolder

description :

Retrieve and show a list of folders on the local docStore FU. Decode and display list if success or indicate cause of failure.

result :

Command OK

example :

```
> DocListFolder;
Command OK (stuDocListFolder)

> DocCreateFolder -ownerName=kqlee -folderComment="example folder";
folderId=0
Command OK (stuDocCreateFolder)

> DocListFolder;
stmDecodeFolderList
folderId=(0) ownerName=(kqlee) folderComment=(example folder) createDateTime=(Tue Oct 26 01:47:57 1999) usedSize=(0)
freeSize=(0) numberOfDocuments=(0)
Command OK (stuDocListFolder)
```

DocListFolderDoc

syntax :

DocListFolderDoc *-folderId=<int>*

description :

Retrieve a list of documents on the specified folder on the local docStore FU. Decode and display list if success or indicate cause of failure.

parameters:

Parameter	Description
-folderId	Identifier of folder for which documents will be listed

result :

Command OK - list of documents in folder
Command ERROR

example :

```
> DocListFolderDoc -folderId=1;
documentId=(1) ownerName=(kqlee) docComment=(example comment) createDateTime=(Tue Oct 26 01:57:45 1999)
size=(400000) numberOfBlocks=(2)
Command OK (stuDocListFolderDoc)
```

DocMoveDoc

syntax :

DocMoveDoc ***-srcFolderId=<int>*** ***-srcDocumentId=<int>*** *-dstFolderId=<int>*$_{opt}$ *-updateDateTime=<bool>*$_{opt}$

description :

Move a document to a new folder. Display new document Id if success, or indicate cause of failure.

parameters:

Parameter	Description
-srcFolderId	Folder ID of source document
-srcDocumentId	ID of source document
-dstFolderId	Folder ID of destination document (if omitted, then destination is assumed to be the same as the source folder ID)
-updateDateTime	Update (yes or no) the date/time stamp of the document when it is moved. . If omitted, date/time is not updated.

result :

Command OK - ID of new document
Command ERROR

example :

```
> DocListFolder;
stmDecodeFolderList
folderId=(1) ownerName=(kqlee) folderComment=(example folder) createDateTime=(Tue Oct 26 01:54:01 1999) usedSize=(0)
freeSize=(0) numberOfDocuments=(1)
folderId=(2) ownerName=(kqlee) folderComment=(example comment 2) createDateTime=(Tue Oct 26 02:01:14 1999)
usedSize=(0) freeSize=(0) numberOfDocuments=(1)
Command OK (stuDocListFolder)

> DocListFolderDoc -folderId=2;
documentId=(2) ownerName=(kqlee) docComment=(example comment) createDateTime=(Tue Oct 26 01:57:45 1999)
size=(400000) numberOfBlocks=(2)
Command OK (stuDocListFolderDoc)

> DocMoveDoc -srcFolderId=1 -srcDocumentId=1 -dstFolderId=2 ;
documentId=3
Command OK (stuDocMoveDoc)

> DocListFolder;
stmDecodeFolderList
folderId=(1) ownerName=(kqlee) folderComment=(example folder) createDateTime=(Tue Oct 26 01:54:01 1999) usedSize=(0)
freeSize=(0) numberOfDocuments=(0)
folderId=(2) ownerName=(kqlee) folderComment=(example comment 2) createDateTime=(Tue Oct 26 02:01:14 1999)
usedSize=(0) freeSize=(0) numberOfDocuments=(2)
Command OK (stuDocListFolder)

> DocListFolderDoc -folderId=2;
documentId=(2) ownerName=(kqlee) docComment=(example comment) createDateTime=(Tue Oct 26 01:57:45 1999)
size=(400000) numberOfBlocks=(2)
documentId=(3) ownerName=(kqlee) docComment=(example comment) createDateTime=(Tue Oct 26 01:57:45 1999)
size=(400000) numberOfBlocks=(2)
Command OK (stuDocListFolderDoc)
```

DocCopyDoc

syntax :

DocCopyDoc ***-srcFolderId=<int>*** ***-srcDocumentId=<int>*** *-dstFolderId=<int>*$_{opt}$ *-updateDateTime=<bool>*$_{opt}$

description :

Copy a document to a new folder. Display new document Id if success, or indicate cause of failure.

parameters:

Parameter	Description
-srcFolderId	Folder ID of source document
-srcDocumentId	ID of source document
-dstFolderId	Folder ID of destination document (if omitted, then destination is assumed to be the same as the source folder ID)
-updateDateTime	Update (True | False) the date/time stamp of the document when it is copied. If omitted, date/time is not updated.

result :

Command OK - ID of new document
Command ERROR

example :

```
> DocListFolder;
stmDecodeFolderList
folderId=(1) ownerName=(kqlee) folderComment=(example folder) createDateTime=(Tue Oct 26 01:54:01 1999) usedSize=(0)
freeSize=(0) numberOfDocuments=(1)
folderId=(2) ownerName=(kqlee) folderComment=(example comment 2) createDateTime=(Tue Oct 26 02:01:14 1999)
usedSize=(0) freeSize=(0) numberOfDocuments=(0)
Command OK (stuDocListFolder)

> DocCopyDoc -srcFolderId=1 -srcDocumentId=1 -dstFolderId=2 ;
documentId=2
Command OK (stuDocCopyDoc)

> DocListFolder;
stmDecodeFolderList
folderId=(1) ownerName=(kqlee) folderComment=(example folder) createDateTime=(Tue Oct 26 01:54:01 1999) usedSize=(0)
freeSize=(0) numberOfDocuments=(1)
folderId=(2) ownerName=(kqlee) folderComment=(example comment 2) createDateTime=(Tue Oct 26 02:01:14 1999)
usedSize=(0) freeSize=(0) numberOfDocuments=(1)
Command OK (stuDocListFolder)

> DocListFolderDoc -folderId=2;
documentId=(2) ownerName=(kqlee) docComment=(example comment) createDateTime=(Tue Oct 26 01:57:45 1999)
size=(400000) numberOfBlocks=(2)
Command OK (stuDocListFolderDoc)
```

DocDeleteDoc

syntax :

DocDeleteDoc *-folderId=<int> -documentId=<int>*

description :

Delete a document from the local docstore FU. Indicate success or cause of failure.

parameters:

Parameter	Description
-folderId	Folder ID of the document to be deleted
-documentId	Document ID of the document to be deleted

result :

Command OK
Command ERROR

example :

```
> DocListFolderDoc -folderId=2;
documentId=(2) ownerName=(kqlee) docComment=(example comment) createDateTime=(Tue Oct 26 01:57:45 1999)
size=(400000) numberOfBlocks=(2)
documentId=(3) ownerName=(sms) docComment=(new comment) createDateTime=(Tue Oct 26 01:57:45 1999)
size=(400000) numberOfBlocks=(2)
Command OK (stuDocListFolderDoc)

> DocDeleteDoc -folderId=2 -documentId=3;
Command OK (stuDocDeleteDoc)

> DocListFolderDoc -folderId=2;
documentId=(2) ownerName=(kqlee) docComment=(example comment) createDateTime=(Tue Oct 26 01:57:45 1999)
size=(400000) numberOfBlocks=(2)
Command OK (stuDocListFolderDoc)
```

DocChangeDocDescription

syntax :

DocChangeDocDescription ***-folderId=<int>*** ***-documentId=<int>*** *-ownerName=<string>*$_{opt}$ *-docComment=<string>*$_{opt}$

description :

Change description fields of document on the local docStore FU. Indicate success or cause of failure.

parameters:

Parameter	Description
-folderId	ID of folder where document is stored
-documentId	ID of document to be updated
-ownerName	Character string identifying document owner
-docComment	Character string commenting document

result :

Command OK
Command ERROR

example :

```
> DocListFolderDoc -folderId=2;
documentId=(2) ownerName=(kqlee) docComment=(example comment) createDateTime=(Tue Oct 26 01:57:45 1999)
size=(400000) numberOfBlocks=(2)
documentId=(3) ownerName=(kqlee) docComment=(example comment) createDateTime=(Tue Oct 26 01:57:45 1999)
size=(400000) numberOfBlocks=(2)
Command OK (stuDocListFolderDoc)

> DocChangeDocDescription -folderId=2 -documentId=3 -ownerName=sms -docComment="new comment";
Command OK (stuDocChangeDocDesc)

> DocListFolderDoc -folderId=2;
documentId=(2) ownerName=(kqlee) docComment=(example comment) createDateTime=(Tue Oct 26 01:57:45 1999)
size=(400000) numberOfBlocks=(2)
documentId=(3) ownerName=(sms) docComment=(new comment) createDateTime=(Tue Oct 26 01:57:45 1999)
size=(400000) numberOfBlocks=(2)
Command OK (stuDocListFolderDoc)
```

DocCreateFolder

syntax :

DocCreateFolder *-ownerName=<string>$_{opt}$ -folderComment=<string>$_{opt}$*

description :

Create a folder on the local docStore FU.

parameters:

Parameter	Description
-ownerName	Character string identifying folder owner
-folderComment	Character string commenting folder

result :

Command OK - folder ID
Command ERROR

example :

```
> DocListFolder;
Command OK (stuDocListFolder)

> DocCreateFolder -ownerName=kqlee -folderComment="example folder";
folderId=0
Command OK (stuDocCreateFolder)

> DocListFolder;
stmDecodeFolderList
folderId=(0) ownerName=(kqlee) folderComment=(example folder) createDateTime=(Tue Oct 26 01:47:57 1999) usedSize=(0)
freeSize=(0) numberOfDocuments=(0)
Command OK (stuDocListFolder)
```

DocDeleteFolder

syntax :

DocDeleteFolder *-folderId=<int>*

description :

Deletes a folder on the local docStore FU. Folders may only be deleted if there are no documents stored in the folder.

parameters:

Parameter	Description
-folderId	Folder ID of folder to delete

result :

Command SUCCESS
Command ERROR

example :

```
> DocListFolder;
stmDecodeFolderList
folderId=(0) ownerName=(kqlee) folderComment=(example folder) createDateTime=(Tue Oct 26 01:47:57 1999) usedSize=(0)
freeSize=(0) numberOfDocuments=(0)
Command OK (stuDocListFolder)
> DocDeleteFolder -folderId=0;
Command OK (stuDocDeleteFolder)
> DocListFolder;
Command OK (stuDocListFolder)
> DocDeleteFolder -folderId=0;
Command Error (stuDocDeleteFolder):
  sfuErrno   = 0x69, 4675420
  descriptor = sfuDocDeleteFolder.invalid folderId
```

DocChangeFolderDescription

syntax :

DocChangeFolderDescription ***-folderId=<int>*** *-ownerName=<string>$_{opt}$ -folderComment=<string>$_{opt}$*

description :

Change description fields of folder on the local docStore FU. Indicate success or cause of failure.

parameters:

Parameter	Description
-folderId	ID of folder to update
-ownerName	Character string identifying the new folder owner
-folderComment	Character string with new folder description

result :

Command SUCCESS
Command OK

example :

```
> DocListFolder;
stmDecodeFolderList
folderId=(1) ownerName=(kqlee) folderComment=(example folder) createDateTime=(Tue Oct 26 01:54:01 1999) usedSize=(0)
freeSize=(0) numberOfDocuments=(0)
folderId=(2) ownerName=(kqlee) folderComment=(example comment 2) createDateTime=(Tue Oct 26 02:01:14 1999)
usedSize=(0) freeSize=(0) numberOfDocuments=(2)
Command OK (stuDocListFolder)

> DocChangeFolderDescription -folderId=2 -ownerName=sms -folderComment="new comment";
Command OK (stuDocChangeFolderDesc)

> DocListFolder;
stmDecodeFolderList
folderId=(1) ownerName=(kqlee) folderComment=(example folder) createDateTime=(Tue Oct 26 01:54:01 1999) usedSize=(0)
freeSize=(0) numberOfDocuments=(0)
folderId=(2) ownerName=(sms) folderComment=(new comment) createDateTime=(Tue Oct 26 02:01:14 1999) usedSize=(0)
freeSize=(0) numberOfDocuments=(2)
Command OK (stuDocListFolder)
```

Send Fax FU Commands (Sfx)

These commands affect the status and operation of the local Send Fax (sfx) FU. To perform Sfx commands on a remote FU, use the appropriate 'Rmt' commands.

The following are the available local Send Fax FU commands:

SfxDsStatus
SfxDsErrorStatus
SfxListSfxJob
SfxSendSfx

The following Dynamic Status Parameters are defined for [Fax Data Send] Functional Unit.

Dynamic Status Parameter	Query	Event	ID	Description
FaxSendStatus	Yes	Yes	12000	status of FAX equipment at sending side.
FaxSendFreeStorageSize	Yes	No	12001	storage size available for spool.
FaxSendErrorStatus	Yes	No	12002	the detail error status information.

SfxDsStatus

syntax :

SfxDsStatus *-status*=<*enum*>

description :

Set an Sfx status (dynamicStatus id = 12000) on the FU. Send notification to all subscribers of this dynamic status Id.

parameters:

Parameter	Description
-status	Status value to be set. Enumerated value: powerFailure == (0) warmingUp == (1) offline == (2) ready == (3) sending == (4) receiving == (5) error == (6) others == (127)

result :

Command OK

notes :

This command will accept any status value. There is no check for valid enumeration values.

example :

```
> SfxDsStatus -status=2;
Command OK (stuSfxDsStatus)
```

SfxDsErrorStatus

syntax :

SfxDsErrorStatus *-systemError*=<*string*> *-others*=<*string*>

description :

Set an Sfx error status (dynamic status id = 12002) on the FU.

parameters:

Parameter	Description
-systemError	String containing error description
-others	String containing other description of error

result :

Command OK

example :

```
> SfxDsErrorStatus -systemError="example system error" -others="";
Command OK (stuSfxDsErrorStatus)
```

SfxListSfxJob

syntax :

SfxListSfxJob

description :

List Sfx jobs on the job queue. This should return the same output as the equivalent remote command.

result :

Command OK- list of jobs

example :

```
> SfxListSfxJob;
jobHandle=(4096) requesterUserId=() jobStatusCode=(1) dataSize=(400000) numOfJobEntries=(1)
Command OK (stuSfxListSfxJob)
```

SfxSendSfx

syntax :

SfxSendSfx ***-todHandle=<int>*** *-dodHandle=<int>$_{opt}$ -modeOfDataTransfer=<enum>$_{opt}$ -dataSource={<ipaddress$_{slmIp}$><int$_{fuHandle}$>}$_{opt}$ -dataHandle=<int>$_{opt}$ -life=<enum>$_{opt}$ -notificationMode={(<enumlist$_{jobStatusCode}$>) <bool$_{entries}$>}$_{opt}$ -notificationScheme={<ipaddress$_{slmIp}$><int$_{fuHandle}$>}$_{opt}$ -priority=<int>$_{opt}$ -retryCount=<int>$_{opt}$*

description :

Create and queue sendFax command to the local SendFax FU. Any parameters specified in this command will override the defaults (if any) specified in the transfer data object. Many elements, especially nested components, of the sendFax definition must be built using TodSendSfxCreate, TodSendSfxTsinfoSet and TodSendSfxEntryPut commands prior to issuing this command.

parameters:

Parameter	Description
-todHandle	FU unique handle assigned to TOD when it is created. See caution in notes.
-dodHandle	FU unique handle assigned to DOD when it is created. Including a dodHandle will create an additional reference to that DOD. See caution in notes.
-modeOfDataTransfer	Enumerated list – immediate == 0, delayed == 1. If not defined, immediate is assumed.
-dataSource	Address of remote (3rd party) data source if data is not to be transferred from client FU. See caution in notes.
-dataHandle	Handle of remote (3rd party) data if data is not to be transferred from client FU. See caution in notes.
-life	Enumerated list – job == (0), session == (1), persistent == (2)
-notificationMode	Parameter includes a set of Job Status Codes and a boolean value allowing for a request for Job Entry Status. Only a Client FU can receive job status notifications. Enumerated list – see Salutation Architecture Specification, V2.0, Part 2, Page 201 for a listing of job status codes.
-notificationScheme	This parameter allows for notification to an FU other than the one that has initiated the job. Only a Client FU can receive job status notifications.
-priority	Integer (0..100) -- 50 is the normal priority, -- 100 is the highest priority. Valid values are enumerated in the capability attributes on a per FU basis.
-retryCount	FU tries to re-call the remote telephone retryCount times, when the remote telephone was busy.

result :

Command OK
Command ERROR

notes :

CAUTION: A valid data source must be defined for this command to function properly. There are three ways that the data source can be referenced:

1 – Reference a DOD with –dodHandle parameter.
2 – Reference a TOD with –todHandle parameter. The TOD must then reference a DOD.
3 – Reference a 3rd party data source with –dataSource and –dataHandle parameters.

example :

```
> SfxSendSfx -todHandle=3 -retryCount=5 -priority=100;
jobHandle=4096
Command OK (stuSfxSendSfx)
```

Extended Fax FU (Efx)

These commands affect the status and operation of the local Extended Fax (efx) FU. To perform Efx commands on a remote FU, use the appropriate 'Rmt' commands.

The following are the available local Extended Fax FU commands:

- EfxRegisterUser
- EfxUnregisterUser
- EfxChangePassword
- EfxListAllUserId
- EfxDsStatus
- EfxDsErrorStatus
- EfxListEfxJob
- EfxSendEfx
- EfxQueryReadInformation

The following Dynamic Status Parameters are defined for [Fax Data] Functional Unit.

Dynamic Status Parameter	Query	Event	ID	Description
FaxStatus	Yes	Yes	13000	status of FAX equipment at sending side.
FaxFreeStorageSize	Yes	No	13001	storage size available for spool.
FaxErrorStatus	Yes	No	13002	the detail error status information.
NumOfSubscribers	Yes	No	13003	the number of current subscribers

EfxRegisterUser

syntax :

EfxRegisterUser ***-user={ <string*** $_{userID}$ ***>*** $<string_{password}>_{opt}$ ***}***

description :

Register a user on a Fax FU. You do not have to be the “Administrator” to use this local command. The remote equivalent RmtRegisterUser requires Administrator privileges.

parameters:

Parameter	Description
-user	UserID / Password for session authentication

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
>  EfxRegisterUser -user={ "Bob Smith" 7gj4ss } ;
Command OK (stuEfxRegisterUser)
```

EfxUnregisterUser

syntax :

EfxUnregisterUser *-userId=<string> -forceDelete=<bool>*

description :

Unregister a user on a Fax FU. You do not have to be the “Administrator” to use this local command. The remote equivalent RmtUnregisterUser requires Administrator privileges.

parameters:

Parameter	Description
-userId	User ID string
-forceDelete	Force delete of user and user-data even if user is not notified

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> EfxUnregisterUser -userId="Bob Smith" -forceDelete=true;
Command OK (stuEfxRegisterUser)
```

EfxChangePassword

syntax :

EfxChangePassword *-**userId**=<**string**>* *-**oldPassword**=<**string**>* *-newPassword=<string>*$_{opt}$

description :

Change a user password on a Fax FU. This command allows you to change any user's password.

parameters:

Parameter	Description
-userId	A registered user name
-oldPassword	Password string that is currently registered with an Extended Fax FU
-newPassword	New password string to be registered with an Extended Fax FU

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> EfxRegisterUser -user={"Bob Smith" mypass};
Command OK (stuEfxRegisterUser)

> EfxChangePassword -userId="Bob Smith" -oldPassword=mypass -newPassword="passcode 55";
Command OK (stuEfxChangePassword)
```

EfxListAllUserId

syntax :

EfxListAllUserId

description :

Retrieve a list of registered users on a Fax FU. Decode and display list if success, or indicate cause of failure.

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	RDATACOMPLETE	Data transfer from remote FU was completed, display decoded list
	RDATAABORT	Data transfer from remote FU was aborted
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> efxlistalluserid;
userId=(dennis)
userId=(Bob Smith)
userId=(Hiroshi)
Command OK (stuEfxListAllUserID)
```

EfxDsStatus

syntax :

EfxDsStatus *-status*=<*enum*>

description :

Set an Efx status (dynamic status id = 13000) on the FU. Send notifications (notifyEvent) to all subscribers of this dynamic status id.

parameters:

Parameter	Description
-status	Status value to be set. Enumerated value: powerFailure == (0) warmingUp == (1) offline == (2) ready == (3) sending == (4) receiving == (5) error == (6) others == (127)

result :

Command OK

notes:

The dynamic status can be set with this command. But remote verification is not supported at this time for Efx FU's.

example :

```
> EfxDsStatus -status=4;
Command OK (stuEfxDsStatus)
```

EfxDsErrorStatus

syntax :

EfxDsErrorStatus *-systemError*=<*string*> *-others*=<*string*>

description :

Set an Efx error status (dynamic status id = 13002) on the FU.

parameters:

Parameter	Description
-systemError	String containing error description
-others	String containing other description of error

result :

Command OK

notes :

This event may not be subscribed by a client.

example :

```
> EfxDsErrorStatus -systemError="no dial tone" -others="no power";
Command OK (stuEfxDsErrorStatus)
```

EfxListEfxJob

syntax :

EfxListEfxJob

description :

List Efx jobs on the job queue.

result :

Command OK - list (possibly empty) of extended fax jobs on queue

example :

```
> EfxListEfxJob;
jobHandle=(4096) requesterUserId=() jobStatusCode=(1) issueTime=(11 August) comment=(mycomment) dataSize=(400000)
numOfJobEntries=(2) requestPriority=(100)
jobHandle=(4097) requesterUserId=() jobStatusCode=(1) issueTime=(11 August) comment=(mycomment) dataSize=(400000)
numOfJobEntries=(3) requestPriority=(100)
Command OK (stuEfxListEfxJob)
```

EfxSendEfx

syntax :

EfxSendEfx ***-todHandle=<int>*** *-dodHandle=<int>$_{opt}$ -modeOfDataTransfer=<enum>$_{opt}$ -dataSource={<ipaddress$_{slmIp}$><int$_{fuHandle}$>}$_{opt}$ -dataHandle=<int>$_{opt}$ -notificationMode={(<enumlist$_{jobStatusCode}$>) <bool$_{entries}$>}$_{opt}$ -notificationScheme={<ipaddress$_{slmIp}$><int$_{fuHandle}$>}$_{opt}$ -retryCount=<int>$_{opt}$ -retryInterval=<int>$_{opt}$ -queryInterval=<int>$_{opt}$ -coverSheetGeneration=<bool>$_{opt}$ -pageHeaderGeneration=<bool>$_{opt}$ -priority=<int>$_{opt}$*

description :

Create and queue a sendEfx command to the local Efx FU. A TOD must be specified. Any parameters specified in this command will override the defaults (if any) specified in the transfer data object. Many elements, especially nested components, of the sendEfx definition must be built using todEfx commands prior to issuing this command.

parameters:

Parameter	Description
-todHandle	FU unique handle assigned to TOD when it is created. See caution in notes.
-dodHandle	FU unique handle assigned to DOD when it is created. Including a dodHandle will create an additional reference to that DOD. See caution in notes.
-modeOfDataTransfer	Enumerated list – immediate == 0, delayed == 1. If not defined, immediate is assumed.
-dataSource	Address of remote (3rd party) data source if data is not to be transferred from client FU. See caution in notes.
-dataHandle	Handle of remote (3rd party) data if data is not to be transferred from client FU. See caution in notes.
-notificationMode	Parameter includes a set of Job Status Codes and a boolean value allowing for a request for Job Entry Status. Only a Client FU can receive job status notifications. Enumerated list – see Salutation Architecture Specification, V2.0, Part 2, Page 201 for a listing of job status codes.
-notificationScheme	This parameter allows for notification to an FU other than the one that has initiated the job. Only a Client FU can receive job status notifications.
-retryCount	FU tries to re-call the remote telephone retryCount times, when the remote telephone was busy.
-retryInterval	Interval for FU retries to call the remote telephone (expressed in minutes)
-queryInterval	Value for when the client application needs to call to get Read Information. [FaxData] FU calls at “queryInterval” minute intervals.
-coverSheetGeneration	Value for whether a cover sheet will be generated. Boolean – include cover sheet == True, no cover sheet == False
-pageHeaderGeneration	Value for whether a page header will be added to the image. Boolean – include header == True, no header== False
-priority	Integer (0..100) -- 50 is the normal priority, -- 100 is the highest priority. Valid values are enumerated in the capability attributes on a per FU basis.

result :

Command OK
Command ERROR

notes :

CAUTION: A valid data source must be defined for this command to function properly. There are three ways that the data source can be referenced:

1 – Reference a DOD with –dodHandle parameter.
2 – Reference a TOD with –todHandle parameter. The TOD must then reference a DOD.
3 – Reference a 3rd party data source with –dataSource and –dataHandle parameters.

example :

```
> EfxSendEfx -todHandle=4  -priority=100;
jobHandle[0]=4098
jobHandle[1]=4099
Command OK (stuEfxSendEfx)
```

EfxQueryReadInformation

syntax :

EfxQueryReadInformation ***-todHandle=<int>*** *-notificationMode={(<enumlist$_{jobStatusCode}$>) <bool$_{entries}$>}$_{opt}$ -notificationScheme={<ipaddress$_{slmIp}$><int$_{fuHandle}$>}$_{opt}$ -retryCount=<int>$_{opt}$ -retryInterval=<int>$_{opt}$ -scheduledDateTime=<string>$_{opt}$ -scheduledAfterTime=<string>$_{opt}$ -priority=<int>$_{opt}$*

description :

Create and queue a queryReadInformation command to the local FU. Any parameters specified in this command will override the defaults (if any) specified in the transfer data object. Many elements, especially those nested components, of the queryReadInformation definition must be built using todEfxQry commands prior to issuing this command.

parameters:

Parameter	Description
-todHandle	FU unique handle assigned to TOD when it is created
-notificationMode	Parameter includes a set of Job Status Codes and a boolean value allowing for a request for Job Entry Status. Only a Client FU can receive job status notifications. Enumerated list – see Salutation Architecture Specification, V2.0, Part 2, Page 201 for a listing of job status codes.
-notificationScheme	This parameter allows for notification to an FU other than the one that has initiated the job. Only a Client FU can receive job status notifications.
-retryCount	FU tries to re-call the remote telephone retryCount times, when the remote telephone was busy.
-retryInterval	FU tries to re-call the remote telephone at retryInterval minute intervals.
-scheduledDateTime	Time for scheduled future delivery. Expressed in UTCTime format.
-scheduledAfterTime	Time delay before delivery. Expressed in "hh:mm:ss" format.
-priority	Integer (0..100) -- 50 is the normal priority, -- 100 is the highest priority. Valid values are enumerated in the capability attributes on a per FU basis.

result :

Command OK - show new job handle
Command ERROR

example :

```
> EfxQueryReadInformation -todHandle=5 -priority=100;
Command OK (stuEfxQueryReadInformation)
```

Remote Attribute Commands

The following are the available Remote Attribute commands:

RmtGetGlobalAttribute
RmtGetPrivateAttribute
RmtSetPrivateAttribute

RmtGetGlobalAttribute

syntax :

RmtGetGlobalAttribute *-attributeId=( )$_{intset}$*

description :

Get the specified global attribute value from the target FU. Decode and display the value, or indicate the cause of failure.

parameters:

Parameter	Description
-attributeId	Remote FU attribute Id to be shown. See Salutation Architecture Specification, V2.0, Part 2, section 9.0 for a listing of attribute Id's

result :

Command OK	ACK	Acknowledgement received, show attribute Id/value pairs
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtGetGlobalAttribute -attributeId=( 10040 10041 10044 );
3 Attributes
attributeId = 10040  enum-value = 100
attributeId = 10041  enum-value = 0
attributeId = 10044  int-value = 20
Command OK (stuRmtGetGlobalAttribute)
```

notes :

The set of supported Global Attributes for a particular FU type is defined in the Salutation Architecture. The tester must ensure the validity of his chosen attribute id. The Salutation Architecture provides no method for remotely setting global attribute values. Consequently, the value returned in this test must have been set in a vendor specific manner on the server FU.

RmtGetPrivateAttribute

syntax :

RmtGetPrivateAttribute *-attributeId=( )$_{intset}$*

description :

Get the specified global attribute value from the target FU. Decode and display the value, or indicate the cause of failure.

parameters:

Parameter	Description
-attributeId	Remote FU attribute Id to be shown. See Salutation Architecture Specification, V2.0, Part 2, section 9.0 for a listing of attribute Id's

result :

Command OK	ACK	Acknowledgement received, show attribute Id/Value pairs
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtGetPrivateAttribute -attributeId=( 10044 );
1 Attributes
attributeId = 10044 int-value = 37
Command OK (stuRmtGetPrivateAttribute)
```

notes :

The set of supported Global Attributes for a particular FU type is defined in the Salutation Architecture. The tester must ensure the validity of its chosen attribute id. Private attributes for a specific session do not exist until they have been explicitly set.

The only defined type of Private attribute is the DocStore FU ModeOfStore attribute (11010). This command will however allow you to get any attribute that was set as a private attribute with PrivateAttributeSet.

RmtSetPrivateAttribute

syntax :

RmtSetPrivateAttributeInt ***-attributeId=<int>*** *-attributeValue=<int>*$_{opt}$

RmtSetPrivateAttributeEnum ***-attributeId=<int>*** *-attributeValue=<enum>*$_{opt}$

RmtSetPrivateAttributeBool ***-attributeId=<int>*** *-attributeValue=<bool>*$_{opt}$

RmtSetPrivateAttributeString ***-attributeId=<int>*** *-attribueValue=<string>*$_{opt}$

description :

Set the specified private attribute to the indicated value. If no value is indicated, the attribute value should be removed from the session's private attribute list.

parameters:

Parameter	Description
-attributeId	Remote FU attribute Id to be shown. See Salutation Architecture Specification, V2.0, Part 2, section 9.0 for a listing of attribute Id's
-attributeValue	Value that attribute will be set to.

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtSetPrivateAttributeInt -attributeId=10044 -attributeValue=37;
ACK
Command OK (stuRmtSetPrivateAttributeInt)
```

notes :

The set of supported Global Attributes for a particular FU type is defined in the Salutation Architecture. The tester must ensure the validity of his chosen attribute id. Private attributes for a specific session do not exist until they have been explicitly set.

The only defined type of Private attribute is the DocStore FU ModeOfStore attribute (11010). This command will however allow you to set any attribute of any type.

Remote Dynamic Status Commands

The following are the available Remote Dynamic Status commands:

RmtQueryDynamicStatus
RmtSubscribeEvent
RmtUnsubscribeEvent

The following Dynamic Status Parameters are defined for [Print] Functional Unit.

Dynamic Status Parameter	Query	Event	ID	Description
PrinterOperationStatus	Yes	Yes	10000	status of printing equipment.
PrinterErrorDetail	Yes	No	10001	detail error information of equipment's.
FreeStorageSize	Yes	No	10002	available storage size, K Byte.
PrinterPaperInputTray	Yes	No	10003	status of paper size and direction in each input tray.
ListExcerptPrintJob	Yes	Yes	10004	lists a excerpt from print job descriptions

The following Dynamic Status Parameters are defined for [Document Storage] Functional Unit.

Dynamic Status Parameter	Query	Event	ID	Description
FreeStorageSize	Yes	No	11000	available storage size.
OperatorIntervention	No	Yes	11001	a warning message to operator or administrator to request human intervention
OperatorInformation	No	Yes	11002	an informational message to operator or administrator

The following Dynamic Status Parameters are defined for [Fax Data Send] Functional Unit.

Dynamic Status Parameter	Query	Event	ID	Description
FaxSendStatus	Yes	Yes	12000	status of FAX equipment at sending side.
FaxSendFreeStorageSize	Yes	No	12001	storage size available for spool.
FaxSendErrorStatus	Yes	No	12002	the detail error status information.

The following Dynamic Status Parameters are defined for [Fax Data] Functional Unit.

Dynamic Status Parameter	Query	Event	ID	Description
FaxStatus	Yes	Yes	13000	status of FAX equipment at sending side.
FaxFreeStorageSize	Yes	No	13001	storage size available for spool.
FaxErrorStatus	Yes	No	13002	the detail error status information.
NumOfSubscribers	Yes	No	13003	the number of current subscribers

RmtQueryDynamicStatus

syntax :

RmtQueryDynamicStatus *-dynamicStatusId=<int>*

description :

Retrieve a dynamic status from the target FU. Decode and display status or indicate cause of failure.

parameters:

Parameter	Description
-dynamicStatusId	ID number of the dynamic status being queried

result :

Command OK	ACK	Acknowledgement received, display decoded status
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

notes :

The set of dynamic status id's supported by a particular FU is specified by the Salutation Architecture, and further limited by those declared in the *'dynamic status supported'* capability attribute.

example :

```
> RmtQueryDynamicStatus -dynamicStatusId=10000;
Command OK (stuRmtQueryDynamicStatus)
ACK()
printerStatusCode = 0 1
```

RmtSubscribeEvent

syntax :

RmtSubscribeEvent ***-dynamicStatusId=(enumlist)*** ***-life=<enum>*** *-checkInterval=<int>$_{opt}$*

description :

Subscribe Client FU to notification of changes in the specified dynamic status id's. Display the assigned subscription handle or indicate cause of failure.

parameters:

Parameter	Description
-dynamicStatusId	ID number of the dynamic status being queried. Must be enclosed by parentheses
-life	Enumerated list – job == (0), session == (1), persistent == (2)
-checkInterval	Integer defining number of seconds between checks to see if remote FU is still active

result :

Command OK	ACK	Acknowledgement received, display subscription handle
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

notes :

The set of dynamic status id's supported by a particular FU and subscribeable by a client is specified by the Salutation Architecture, and further limited by those declared in the *'dynamic status supported'* capability attribute.

example :

Clt Dialog Window
> RmtSubscribeEvent -dynamicStatusId=(10000) -life=2;
ACK
Arg1:int-value = 1
Command OK (stuRmtSubscribeEvent)

Prn Dialog Window
> PrnDsOperationStatus -notoner=true -nopaper=true;
Command OK (stuPrnDsOperationStatus)

Clt Notification Window
defaultNcb()...
notifyEvent(sbsHandle=1,id=10000)
printerStatusCode = 0 1

RmtUnsubscribeEvent

syntax :

RmtUnsubscribeEvent *-subscriptionHandle=<int>*

description :

Delete a previously established subscription. Indicate success or cause of failure.

parameters:

Parameter	Description
-subscriptionHandle	Dynamic status subscription handle given when the subscription was initiated.

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

notes :

The set of dynamic status id's supported by a particular FU and subscribeable by a client is specified by the Salutation Architecture, and further limited by those declared in the *'dynamic status supported'* capability attribute.

example :

```
> RmtUnsubscribeEvent -subscriptionHandle=1;
ACK
Command OK (stuRmtUnsubscribeEvent)
```

Remote Job Commands

The following are the available remote job commands:

- RmtQueryJobStatus
- RmtCancelJob
- RmtSuspendJob
- RmtResumeJob
- RmtChangeJobAttribute
- RmtStartMonitorJobStatus
- RmtCancelMonitorJobStatus
- RmtFreeJobHandle

RmtQueryJobStatus

syntax :

RmtQueryJobStatus *-jobHandle=<int>*

description :

Retrieve the current status of the specified job on the target FU job queue. Display status or indicate cause of failure.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.

result :

Command OK	ACK	Acknowledgement received, display status
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtSuspendJob -jobHandle=4096;
ACK
Command OK (stuRmtSuspendJob)
> RmtQueryJobStatus -jobHandle=4096;
ACK
Arg1:int-value = 3
Command OK (stuRmtQueryJobStatus)
```

RmtCancelJob

syntax :

RmtCancelJob *-jobHandle=<int> -abort=<bool>*

description :

Cancel processing of the specified job on the target FU job queue. Indicate success or cause of failure.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.
-abort	Determines if a job can be aborted when the job is already being executed. Boolean – True == job is canceled if queued or being executed False == job is canceled only if execution has not started

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

notes :

A job may be set to this state regardless of its former state. This is under control of the user - though the architecture does define a progression (sequence) of valid states, this is not enforced by this tool.

example :

```
> RmtCancelJob -jobHandle=4098 -abort=true;
ACK
Command OK (stuRmtCancelJob)

> RmtQueryJobStatus -jobHandle=4098;
ACK
Arg1:int-value = 7
Command OK (stuRmtQueryJobStatus)
```

RmtSuspendJob

syntax :

RmtSuspendJob *-jobHandle=<int>*

description :

Suspend a job on the FU job queue. Indicate success or cause of failure.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtSuspendJob -jobHandle=4096;
ACK
Command OK (stuRmtSuspendJob)
> RmtQueryJobStatus -jobHandle=4096;
ACK
Arg1:int-value = 3
Command OK (stuRmtQueryJobStatus)
```

RmtResumeJob

syntax :

RmtResumeJob *-jobHandle=<int>*

description :

Resume a previously suspended job on the FU job queue. Indicate success or cause of failure.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtQueryJobStatus -jobHandle=4096;
ACK
Arg1:int-value = 3
Command OK (stuRmtQueryJobStatus)
> RmtResumeJob -jobHandle=4096;
ACK
Command OK (stuRmtResumeJob)
> RmtQueryJobStatus -jobHandle=4096;
ACK
Arg1:int-value = 1
Command OK (stuRmtQueryJobStatus)
```

RmtChangeJobAttribute

syntax :

RmtChangeJobAttributeInt *-jobHandle=<int> -attributeId=<int> -attributeValue=<int>*

RmtChangeJobAttributeEnum *-jobHandle=<int> -attributeId=<int> -attributeValue=<enum>*

RmtChangeJobAttributeBool *-jobHandle=<int> -attributeId=<int> -attributeValue=<bool>*

RmtChangeJobAttributeString *-jobHandle=<int> -attributeId=<int> -attributeValue=<string>*

description :

Change or set an attribute of a job on the FU job queue. Indicate success or cause of failure.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.
-attributeId	Id of the job attribute.
-attributeValue	Value of the attribute

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

Prn Dialog Window
> JobAttributeGet -jobHandle=4096 -attributeId=10040;
enum-value = 50
Command OK (stuJobAttributeGet)

Client Dialog Window
> RmtChangeJobAttributeEnum -jobHandle=4096 -attributeId=10040 -attributeValue=100;
ACK
Command OK (stuRmtChangeJobAttributeEnum)

Prn Dialog Window
> JobAttributeGet -jobHandle=4096 -attributeId=10040;
enum-value = 100
Command OK (stuJobAttributeGet)

RmtStartMonitorJobStatus

syntax :

RmtStartMonitorJobStatus *-jobHandle=<int> -notificationMode={(<enumlist$_{jobStatusCode}$>) <bool$_{entries}$>}$_{opt}$*

description :

Subscribe to notification of change in job status. Display subscription handle if success or indicate cause of failure.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.
-notificationMode	Parameter includes a set of Job Status Codes and a boolean value allowing for a request for Job Entry Status. Only a Client FU can receive job status notifications. Enumerated list – see Salutation Architecture Specification, V2.0, Part 2, Page 201 for a listing of job status codes.

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

notes :

This command can be issued by any FU type. However, only a Client will receive notifications. All other FU's will NACK the remote FU when it attempts to send the job status.

example :

Client Dialog Window
> **RmtStartMonitorJobStatus -jobHandle=4096 -notificationMode={ (3) false };**
ACK
Command OK (stuRmtStartMonitorJobStatus)
> RmtSuspendJob -jobHandle=4096;
ACK
Command OK (stuRmtSuspendJob)

Client Notification Window
defaultNcb()...
notifyJobStatus(jobHandle=4096,jobStatusCode=3)

RmtCancelMonitorJobStatus

syntax :

RmtCancelMonitorJobStatus *-jobHandle=<int>*

description :

Cancel a previously established job status notification subscription. Indicate success or cause of failure.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtCancelMonitorJobStatus -jobHandle=4096;
ACK
Command OK (stuRmtCancelMonitorJobStatus)
```

RmtFreeJobHandle

syntax :

RmtFreeJobHandle *-jobHandle=<int>*

description :

Free a job handle of a (life=persistent (2)) completed job from the job queue. Indicate success or cause of failure.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtPrint -todHandle=1 -life=2;
ACK
Arg1:int-value = 4097
Command OK (stuRmtPrint)
> RmtListPrintJob;
listRDataComplete(4)
stmDecodePrintJobList
jobHandle=(4096) requesterUserId=() jobStatusCode=(3) dataSize=(400000) printPriority=(100) queuedTime=(Sat Oct 30
04:26:05 1999) printFileName=(printFile)
jobHandle=(4097) requesterUserId=() jobStatusCode=(0) dataSize=(400000) printPriority=(50) queuedTime=(Sat Oct 30
04:39:18 1999) printFileName=(printFile)
Command OK (stuRmtListPrintJob)
> RmtFreeJobHandle -jobHandle=4097;
ACK
Command OK (stuRmtFreeJobHandle)
> RmtListPrintJob;
listRDataComplete(6)
stmDecodePrintJobList
jobHandle=(4096) requesterUserId=() jobStatusCode=(3) dataSize=(400000) printPriority=(100) queuedTime=(Sat Oct 30
04:26:05 1999) printFileName=(printFile)
Command OK (stuRmtListPrintJob)
```

Remote Job Entry Commands

The following are the available local Extended Fax FU commands:

RmtQueryJobEntryStatus
RmtCancelJobEntry
RmtSuspendJobEntry
RmtResumeJobEntry
RmtChangeJobEntryAttribute

RmtQueryJobEntryStatus

syntax :

RmtQueryJobEntryStatus *-jobHandle=<int> -jobEntryId=<int>*

description :

Retrieve the current status of the specified job entry on the target FU job queue. Display status or indicate cause of failure.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.
-jobEntryId	Unique Id assigned to Job Entry

result :

Command OK	ACK	Acknowledgement received, display job entry status
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtQueryJobEntryStatus -jobHandle=4097 -jobEntryId=2;
ACK
Arg1:int-value = 1
Command OK (stuRmtQueryJobEntryStatus)
```

RmtCancelJobEntry

syntax :

RmtCancelJobEntry *-jobHandle=<int> -jobEntryId=<int> -abort=<bool>*

description :

Cancel processing of the specified job entry on the target FU job queue. Indicate success or cause of failure.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.
-jobEntryId	Unique Id assigned to Job Entry
-abort	Determines if a job entry can be cancelled when the job entry is already being executed. Boolean – True == job entry is canceled if queued or being executed False == job entry is canceled only if execution has not started

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtCancelJobEntry -jobHandle=4097 -jobEntryId=2 -abort=true;
ACK
Command OK (stuRmtCancelJobEntry)
```

RmtSuspendJobEntry

syntax :

RmtSuspendJobEntry *-jobHandle=<int> -jobEntryId=<int>*

description :

Suspend a job entry on the FU job queue. Indicate success or cause of failure.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.
-jobEntryId	Unique Id assigned to Job Entry

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtSuspendJobEntry -jobHandle=4097 -jobEntryId=1;
ACK
Command OK (stuRmtSuspendJobEntry)
```

RmtResumeJobEntry

syntax :

RmtResumeJobEntry *-jobHandle=<int> -jobEntryId=<int>*

description :

Resume a previously suspended job entry on the FU job queue. Indicate success or cause of failure.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.
-jobEntryId	Unique Id assigned to Job Entry

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtResumeJobEntry -jobHandle=4097 -jobEntryId=1;
ACK
Command OK (stuRmtResumeJobEntry)
```

RmtChangeJobEntryAttribute

syntax :

RmtChangeJobEntryAttributeInt *-jobHandle=<int> -jobEntryId=<int> -attributeId=<int> -attributeValue=<int>*
RmtChangeJobEntryAttributeEnum *-jobHandle=<int> -jobEntryId=<int> -attributeId=<int> -attributeValue=<enum>*
RmtChangeJobEntryAttributeBool *-jobHandle=<int> -jobEntryId=<int> -attributeId=<int> -attributeValue=<bool>*
RmtChangeJobEntryAttributeString *-jobHandle=<int> -jobEntryId=<int> -attributeId=<int> -attributeValue=<string>*

description :

Change or set an attribute of a job entry on the FU job queue. Indicate success or cause of failure.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.
-jobEntryId	Unique Id assigned to Job Entry
-attributeId	Id of the job attribute.
-attributeValue	Value of the attribute

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtChangeJobEntryAttributeInt -jobHandle=4097 -jobEntryId=2 -attributeId=12032 -
attributeValue=12;
Command OK (stuRmtChangeJobEntryAttributeInt)
```

Remote Data Transfer Commands

The following are the available local Remote Data Transfer commands:

RmtRequestDataTransfer

RmtRequestDataTransfer

syntax :

RmtRequestDataTransfer *-dataHandle=<int>*$_{opt}$

description :

Retrieve data from a remote FU. Primary usage is to retrieve a data object which has been placed in a FU's export pool by 'RmtRetrieveDoc' command with –export=true.

parameters:

Parameter	Description
-dataHandle	Handle of data that is stored in an export pool of a remote FU.

result :

Command OK	NACK	Negative Acknowledgement Received
	RDATACOMPLETE	Data transfer from remote FU was completed - display dodHandle
	RDATAABORT	Data transfer from remote FU was aborted
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> rmtRetrieveDoc -folderId=0 -documentId=1 -export=true;
ACK
Arg1:int-value = 2
Command OK (stuRmtRetrieveDoc)
> RmtRequestDataTransfer -dataHandle=2;
RDataComplete - dodHandle=4097
Command OK (stuRmtRequestDataTransfer)
> DodShow -dodHandle=4097;
DODNODE (4097 ) (dataHandle=4097,size=80,blkcnt=1,bufcnt=9,refCnt=1)
Command OK (stuDodShow)
```

Remote Print FU Commands

The following are the available Remote Print FU commands:

RmtListPrintJob
RmtPrint

RmtListPrintJob

syntax :

RmtListPrintJob

description :

Retrieve a list of print jobs currently on the FU's print job queue. Decode and display the list or indicate cause of failure.

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	RDATACOMPLETE	Data transfer from remote FU was completed, display decoded list
	RDATAABORT	Data transfer from remote FU was aborted
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtListPrintJob;
listRDataComplete(4)
stmDecodePrintJobList
jobHandle=(4096) requesterUserId=() jobStatusCode=(3) dataSize=(400000) printPriority=(100)
queuedTime=(Sat Oct 30 04:26:05 1999) printFileName=(printFile)
jobHandle=(4097) requesterUserId=() jobStatusCode=(0) dataSize=(400000) printPriority=(50)
queuedTime=(Sat Oct 30 04:39:18 1999) printFileName=(printFile)
Command OK (stuRmtListPrintJob)
```

RmtPrint

syntax :

RmtPrint *-todHandle=<int>$_{opt}$ -dodHandle=<int>$_{opt}$ -modeOfDataTransfer=<enum>$_{opt}$ -dataSource={<ipaddress$_{slmIp}$><int$_{fuHandle}$>}$_{opt}$ -dataHandle=<int>$_{opt}$ -life=<enum>$_{opt}$ -notificationMode={(<enumlist$_{jobStatusCode}$>) <bool$_{entries}$>}$_{opt}$ -notificationScheme={<ipaddress$_{slmIp}$><int$_{fuHandle}$>}$_{opt}$ -paperSize=<enum>$_{opt}$ -resolution=<enum>$_{opt}$ -paperDirection=<enum>$_{opt}$ -copyCount=<int>$_{opt}$ -inputSelect=<enum>$_{opt}$ -outputSelect=<enum>$_{opt}$ -outputBinSelect=<int>$_{opt}$ -duplexModeSelect=<enum>$_{opt}$ -duplexBindingMargin=<int>$_{opt}$ -faceUpModeSelect=<enum>$_{opt}$ -priority=<int>$_{opt}$ -staplingSelect=<int>$_{opt}$ -fileName=<string>$_{opt}$*

description :

Send a print command to a remote FU. If client data is to be transferred, then a dodHandle, or a todHandle with associated data, must be specified. Any parameters specified in this command will override the defaults (if any) specified in the transfer data object. Indicate success or indicate cause of failure.

parameters:

Parameter	Description
-todHandle	FU unique handle assigned to TOD when it is created. See caution in notes.
-dodHandle	FU unique handle assigned to DOD when it is created. Including a dodHandle will create an additional reference to that DOD. See caution in notes.
-modeOfDataTransfer	Enumerated list – immediate == 0, delayed == 1. If not defined, immediate is assumed.
-dataSource	Address of remote (3rd party) data source if data is not to be transferred from client FU. See caution in notes.
-dataHandle	Handle of remote (3rd party) data if data is not to be transferred from client FU. See caution in notes.
-life	Enumerated list – job == (0), session == (1), persistent == (2)
-notificationMode	Parameter includes a set of Job Status Codes and a boolean value allowing for a request for Job Entry Status. Only a Client FU can receive job status notifications. Enumerated list – see Salutation Architecture Specification, V2.0, Part 2, Page 201 for a listing of job status codes.
-notificationScheme	This parameter allows for notification to an FU other than the one that has initiated the job. Only a Client FU can receive job status notifications.
-paperSize	Enumerated list – see Salutation Architecture Specification, V2.0, Part 2, Page 210.
-resolution	Print resolution. Enumerated list – see Salutation Architecture Specification, V2.0, Part 2, Page 208
-paperDirection	Paper direction. Enumerated list – portrait == (1), landscape == (2), other == (127)
-copyCount	Integer defining the number of copies requested – default is 1 copy
-inputSelect	Paper tray input selection. Enumerated list - manualFeed == (0), tray-1 == (1), tray-2 == (2), tray-3 == (3), tray-4 == (4), tray-5 == (5), automaticSelect (FU selects tray by PaperSize) == (126), other == (127)
-outputSelect	Output sortation mode. Enumerated list - standard == (0), collatedSort == (1), stack == (2), nonCollatedSort == (3), other == (127)
-outputBinSelect	Output bin number. Integer corresponding to equipment bin number
-duplexModeSelect	Duplex binding mode. Enumerated list - simplex == (0), left-binding-duplex == (1), right-binding-duplex == (2), top-binding-duplex == (3), other == (127)
-duplexBindingMargin	Margin distance for duplex binding. (0.1mm x duplexBindingMargin)??
-faceUpModeSelect	Enumerated list – faceDown == (1), faceUp == (2), other == (127)
-priority	Integer (0..100) -- 50 is the normal priority, -- 100 is the highest priority. Valid values are enumerated in the capability attributes on a per FU basis.
-staplingSelect	Stapling location. Enumerated list – see Salutation Architecture Specification, V2.0, Part 2, Page 215.
-fileName	File name of print job

result :

Command OK	ACK	Acknowledgement received, display job Handle
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
	SDATAABORT	Data transfer to remote FU was aborted
Command ERROR		

notes :

CAUTION: A valid data source must be defined for this command to function properly. There are three ways that the data source can be referenced:

1 – Reference a DOD with –dodHandle parameter.
2 – Reference a TOD with –todHandle parameter. The TOD must then reference a DOD.
3 – Reference a 3rd party data source with –dataSource and –dataHandle parameters.

This command does not support immediate 3rd party data transfers. For 3rd party data sources, you will need to use delayed data transfers.

Delayed mode for 3rd party data transfer is supported, however the data transfer must be done manually. Data transfer will not be started automatically.

example :

```
> RmtPrint -todhandle=1;
ACK
Arg1:int-value = 4096
Command OK (stuRmtPrint)
```

Remote DocStore FU Commands

The following are the available local Extended Fax FU commands:

- RmtListFolder
- RmtListFolderDoc
- RmtRetrieveDoc
- RmtStoreDoc
- RmtMoveDoc
- RmtCopyDoc
- RmtDeleteDoc
- RmtChangeDocDescription
- RmtCreateFolder
- RmtDeleteFolder
- RmtChangeFolderDescription

RmtListFolder

syntax :

RmtListFolder

description :

Retrieve a list of folders on docStore FU. Decode and display list if success or indicate cause of failure.

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	RDATACOMPLETE	Data transfer from remote FU was completed, display decoded list
	RDATAABORT	Data transfer from remote FU was aborted
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtListFolder;
listRDataComplete(8)
stmDecodeFolderList
folderId=(0) ownerName=(kqlee) folderComment=(new folder) createDateTime=(Sat Oct 30 04:44:27
1999) usedSize=(0) freeSize=(0) numberOfDocuments=(0)
Command OK (stuRmtListFolder)
```

RmtListFolderDoc

syntax :

RmtListFolderDoc *-folderId=<int>*

description :

Retrieve a list of documents on the specified folder in a docStore FU. Decode and display list if success or indicate cause of failure.

parameters:

Parameter	Description
-folderId	Identifier of folder for which documents will be listed

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	RDATACOMPLETE	Data transfer from remote FU was completed, display decoded list
	RDATAABORT	Data transfer from remote FU was aborted
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtListFolderDoc -folderId=0;
listRDataComplete(11)
documentId=(1) ownerName=(doc owner) docComment=(doc comment) createDateTime=(Sat Oct 30 04:46:13 1999)
size=(400000) numberOfBlocks=(1)
Command OK (stuRmtListFolderDoc)
```

RmtRetrieveDoc

syntax :

RmtRetrieveDoc ***-export=<bool>*** ***-folderId=<int>*** ***-documentId=<int>*** *-startBlock=<int>$_{opt}$* *-endBlock=<int>$_{opt}$*

description :

Retrieve a document from a remote FU.

Specify (export=TRUE) if the document is to be placed in the export pool and return a dataHandle for later retrieval using RmtRequestDateTransfer.

Specify (export=FALSE) to immediately transfer the data to the FU issuing the RmtRetrieveDoc command. A dodHandle is also returned.

parameters:

Parameter	Description
-export	Boolean - True == place document to export pool, False == place document into a transfer object definition (TOD)
-folderId	Identifier of folder for which document will be retrieved
-documentId	Document ID of the document to be retrieved
-startBlock	Starting data block. If omitted, the document is retrieved from the first data block.
-endBlock	Ending data block. If omitted, the document is retrieved through the last data block.

result :

Command OK	ACK Acknowledgement received If –export=True, display dataHandle of exported doc
	NACK Negative Acknowledgement Received
	RDATACOMPLETE Data transfer from remote FU was completed If –export=False, display dodHandle of transferred data
	RDATAABORT Data transfer from remote FU was aborted
	XTIMEOUT Transaction to remote FU timed out (no response)
	SVCCLOSE Session to remote FU closed during transaction
Command ERROR	

example :

```
> RmtRetrieveDoc -export=false -folderid=0 -documentId=1;
RDataComplete - dodHandle=4097
Command OK (stuRmtRetrieveDoc)
```

RmtStoreDoc

syntax :

RmtStoreDoc *-todHandle=<int>$_{opt}$ -dodHandle=<int>$_{opt}$ -folderId=<int>$_{opt}$ -dataSource={<ipaddress$_{slmIp}$><int$_{fuHandle}$>}$_{opt}$ -dataHandle=<int>$_{opt}$ -modeOfStore=<enum>$_{opt}$ -ownerName=<string>$_{opt}$ -docComment=<string>$_{opt}$*

description :

Send a storeDoc command to a remote FU. If client data is to be transferred, then a dodHandle, or a todHandle with associated data, must be specified. Any parameters specified in this command will override the defaults (if any) previously specified in the transfer data object definition.

parameters:

Parameter	Description
-todHandle	FU unique handle assigned to TOD when it is created. See caution in notes.
-dodHandle	FU unique handle assigned to DOD when it is created. Including a dodHandle will create an additional reference to that DOD. See caution in notes.
-folderId	ID of folder on remote FU into which the new document will go. If –todHandle parameter is not included or if the referenced TOD does not reference a DOD, then this parameter is required. See caution in notes.
-dataSource	Address of remote (3rd party) data source if data is not to be transferred from client FU. See caution in notes.
-dataHandle	Handle of remote (3rd party) data if data is not to be transferred from client FU. See caution in notes.
-modeOfStore	1 == Non-transparent Mode - transmit and store data as multiple blocks (retain block boundaries as when created) 2 == Transparent Mode - transmit and store data a single block (combine blocks) Overrides remote FU's global or private attribute
-ownerName	Name of new document owner
-docComment	Comment string for new document

result :

Command OK	ACK	Acknowledgement received, display assigned document Id
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
	SDATAABORT	Data transfer to remote FU was aborted
Command ERROR		

notes :

CAUTION: A valid Folder ID must defined for this command to function properly. The Folder ID can either be defined at the time of document creation (with this command) or by a Transfer Object Definition (TOD) which must then be referenced by a TOD Handle in this command.

CAUTION: A valid data source must be defined for this command to function properly. There are three ways that the data source can be referenced:

1 – Reference a DOD with –dodHandle parameter.
2 – Reference a TOD with –todHandle parameter. The TOD must then reference a DOD.
3 – Reference a 3rd party data source with –dataSource and –dataHandle parameters.

example :

```
> RmtStoreDoc -todHandle=2 -folderId=0;
ACK
Arg1:int-value = 1
Command OK (stuRmtStoreDoc)
```

RmtMoveDoc

syntax :

RmtMoveDoc ***-srcFolderId=<int>*** ***-srcDocumentId=<int>*** *-dstFolderId=<int>*$_{opt}$ *-updateDateTime=<bool>*$_{opt}$

description :

Move a document to a new folder. Display new document Id if success, or indicate cause of failure.

parameters:

Parameter	Description
-srcFolderId	Folder ID of source document
-srcDocumentId	ID of source document
-dstFolderId	Folder ID of destination document (if omitted, then destination is assumed to be the same as the source folder ID)
-updateDateTime	Update (yes or no) the date/time stamp of the document when it is moved. . If omitted, date/time is not updated.

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtMoveDoc -srcFolderId=1 -srcDocumentId=2 -dstFolderId=0;
ACK
Arg1:int-value = 3
Command OK (stuRmtMoveDoc)
> RmtListFolderDoc -folderId=0;
listRDataComplete(19)
documentId=(1) ownerName=(sms) docComment=(doc comment) createDateTime=(Sat Oct 30
04:46:13 1999) size=(400000) numberOfBlocks=(1)
documentId=(3) ownerName=(sms) docComment=(doc comment) createDateTime=(Sat Oct 30
04:46:13 1999) size=(400000) numberOfBlocks=(1)
Command OK (stuRmtListFolderDoc)
```

RmtCopyDoc

syntax :

RmtCopyDoc ***-srcFolderId=<int>*** ***-srcDocumentId=<int>*** *-dstFolderId=<int>*$_{opt}$ *-updateDateTime=<bool>*$_{opt}$

description :

Copy a document to a new folder. Display new document Id if success, or indicate cause of failure.

parameters:

Parameter	Description
-srcFolderId	Folder ID of source document
-srcDocumentId	ID of source document
-dstFolderId	Folder ID of destination document (if omitted, then destination is assumed to be the same as the source folder ID)
-updateDateTime	Update (yes or no) the date/time stamp of the document when it is copied. If omitted, date/time is not updated.

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtCopyDoc -srcFolderId=0 -srcDocumentId=1 -dstFolderId=1;
ACK
Arg1:int-value = 2
Command OK (stuRmtCopyDoc)
> RmtListFolderDoc -folderId=1;
listRDataComplete(17)
documentId=(2) ownerName=(sms) docComment=(doc comment) createDateTime=(Sat Oct 30 04:46:13 1999) size=(400000)
numberOfBlocks=(1)
Command OK (stuRmtListFolderDoc)
```

RmtDeleteDoc

syntax :

RmtDeleteDoc *-folderId=<int> -documentId=<int>*

description :

Delete a document to from a docstore FU. Indicate success or cause of failure.

parameters:

Parameter	Description
-folderId	Folder ID of the document to be deleted
-documentId	Document ID of the document to be deleted

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtDeleteDoc -folderId=0 -documentId=3;
ACK
Command OK (stuRmtDeleteDoc)
> RmtListFolderDoc -folderId=0;
listRDataComplete(23)
documentId=(1) ownerName=(sms) docComment=(doc comment) createDateTime=(Sat Oct 30 04:46:13 1999) size=(400000)
numberOfBlocks=(1)
Command OK (stuRmtListFolderDoc)
```

RmtChangeDocDescription

syntax :

RmtChangeDocDescription ***-folderId=<int>*** ***-documentId=<int>*** *-ownerName=<string>*$_{opt}$ *-docComment=<string>*$_{opt}$

description :

Change description fields of document on a docStore FU. Indicate success or cause of failure.

parameters:

Parameter	Description
-folderId	ID of folder where document is stored
-documentId	ID of document to be updated
-ownerName	Character string identifying document owner
-docComment	Character string commenting document

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtChangeDocDesc -folderId=0 -documentId=1 -ownerName=sms;
ACK
Command OK (stuRmtChangeDocDesc)
> RmtListFolderDoc -folderId=0;
listRDataComplete(13)
documentId=(1) ownerName=(sms) docComment=(doc comment) createDateTime=(Sat Oct 30 04:46:13 1999) size=(400000)
numberOfBlocks=(1)
Command OK (stuRmtListFolderDoc)
```

RmtCreateFolder

syntax :

RmtCreateFolder *-ownerName=<string>$_{opt}$ -folderComment=<string>$_{opt}$*

description :

Create a new folder on a remote docStore FU.

parameters:

Parameter	Description
-ownerName	Character string identifying folder owner
-folderComment	Character string commenting folder

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtCreateFolder -ownerName=kqlee -folderComment="new folder";
ACK
Arg1:int-value = 0
Command OK (stuRmtCreateFolder)
```

RmtDeleteFolder

syntax :

RmtDeleteFolder *-folderId=<int>*

description :

Delete a folder on a remote docStore FU.

parameters:

Parameter	Description
-folderId	Folder ID of folder to delete

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtListFolder;
listRDataComplete(14)
stmDecodeFolderList
folderId=(0) ownerName=() folderComment=() createDateTime=(Sat Nov 13 16:20:20 1999) usedSize=(0) freeSize=(0)
numberOfDocuments=(0)
folderId=(1) ownerName=() folderComment=() createDateTime=(Sat Nov 13 16:20:38 1999) usedSize=(0) freeSize=(0)
numberOfDocuments=(0)
Command OK (stuRmtListFolder)

> RmtDeleteFolder -folderId=1;
ACK
Command OK (stuRmtDeleteFolder)

> RmtListFolder;
listRDataComplete(16)
stmDecodeFolderList
folderId=(0) ownerName=() folderComment=() createDateTime=(Sat Nov 13 16:20:20 1999) usedSize=(0) freeSize=(0)
numberOfDocuments=(0)
Command OK (stuRmtListFolder)
```

RmtChangeFolderDescription

syntax :

RmtChangeFolderDescription ***-folderId=<int>*** *-ownerName=<string>$_{opt}$ -folderComment=<string>$_{opt}$*

description :

Change description fields of folder on a docStore FU. Indicate success or cause of failure

parameters:

Parameter	Description
-folderId	ID of folder to update
-ownerName	Character string identifying the new folder owner
-folderComment	Character string with new folder description

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtChangeFolderDesc -folderId=0 -folderComment="changed comment";
ACK
Command OK (stuRmtChangeFolderDesc)
> RmtListFolder;
listRDataComplete(25)
stmDecodeFolderList
folderId=(0) ownerName=(kqlee) folderComment=(changed comment) createDateTime=(Sat Oct 30 04:44:27 1999)
usedSize=(0) freeSize=(0) numberOfDocuments=(1)
folderId=(1) ownerName=(sms) folderComment=() createDateTime=(Sat Oct 30 04:53:59 1999) usedSize=(0) freeSize=(0)
numberOfDocuments=(0)
Command OK (stuRmtListFolder)
```

Remote FaxSend FU Commands

The following are the available Remote FaxSend FU commands:

RmtListSfxJob
RmtSendSfx

RmtListSfxJob

syntax :

RmtListSfxJob

description :

Retrieve a list of fax jobs currently on the FU's fax job queue. Decode and display the list or indicate cause of failure.

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	RDATACOMPLETE	Data transfer from remote FU was completed, display decoded list
	RDATAABORT	Data transfer from remote FU was aborted
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtListSfxJob;
listRDataComplete(11)
jobHandle=(4096) requesterUserId=() jobStatusCode=(1) dataSize=(100) numOfJobEntries=(0)
jobHandle=(4097) requesterUserId=() jobStatusCode=(1) dataSize=(80) numOfJobEntries=(2)
Command OK (stuRmtListSfxJob)
```

RmtSendSfx

syntax :

RmtSendSfx ***-todHandle=<int>*** *-dodHandle=<int>$_{opt}$ -modeOfDataTransfer=<enum>$_{opt}$ -dataSource={<ipaddress$_{slmIp}$><int$_{fuHandle}$>}$_{opt}$ -dataHandle=<int>$_{opt}$ -life=<enum>$_{opt}$ -notificationMode={(<enumlist$_{jobStatusCode}$>) <bool$_{entries}$>}$_{opt}$ -notificationScheme={<ipaddress$_{slmIp}$><int$_{fuHandle}$>}$_{opt}$ -priority=<int>$_{opt}$ -retryCount=<int>$_{opt}$*

description :

Send a sendFax command to a remote FU. Any parameters specified in this command will override the defaults (if any) specified in the transfer data object. Many elements, especially nested components, of the sendFax definition must be built using todSfxCreate and todSfx command prior to issuing a sendSfx command.

parameters:

Parameter	Description
-todHandle	FU unique handle assigned to TOD when it is created. See caution in notes.
-dodHandle	FU unique handle assigned to DOD when it is created. Including a dodHandle will create an additional reference to that DOD. See caution in notes.
-modeOfDataTransfer	Enumerated list – immediate == 0, delayed == 1. If not defined, immediate is assumed.
-dataSource	Address of remote (3rd party) data source if data is not to be transferred from client FU. See caution in notes.
-dataHandle	Handle of remote (3rd party) data if data is not to be transferred from client FU. See caution in notes.
-life	Enumerated list – job == (0), session == (1), persistent == (2)
-notificationMode	Parameter includes a set of Job Status Codes and a boolean value allowing for a request for Job Entry Status. Only a Client FU can receive job status notifications. Enumerated list – see Salutation Architecture Specification, V2.0, Part 2, Page 201 for a listing of job status codes.
-notificationScheme	This parameter allows for notification to an FU other than the one that has initiated the job. Only a Client FU can receive job status notifications.
-priority	Integer (0..100) -- 50 is the normal priority, -- 100 is the highest priority. Valid values are enumerated in the capability attributes on a per FU basis.
-retryCount	FU tries to re-call the remote telephone retryCount times, when the remote telephone was busy.

result :

Command OK	ACK	Acknowledgement received, display job Handle
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
	SDATAABORT	Data transfer to remote FU was aborted
Command ERROR		

notes :

CAUTION: A valid data source must be defined for this command to function properly. There are three ways that the data source can be referenced:

1 – Reference a DOD with –dodHandle parameter.
2 – Reference a TOD with –todHandle parameter. The TOD must then reference a DOD.
3 – Reference a 3rd party data source with –dataSource and –dataHandle parameters.

example :

```
> RmtSendSfx -dodHandle=10;
ACK
Arg1:int-value = 4098
Command OK (stuRmtSendSfx)
```

Remote Extended Fax FU Commands

The following are the available local Extended Fax FU commands:

- RmtRegisterUser
- RmtUnregisterUser
- RmtChangePassword
- RmtListAllUserId
- RmtListEfxJob
- RmtSubscribeEfxEvent
- RmtUnsubscribeEfxEvent
- RmtRetrieveEfxData
- RmtRetrieveEfxDocId
- RmtPrintEfxData
- RmtQuerySentEfx
- RmtQueryEfxHistory
- RmtQueryReadInformation
- RmtInformRead
- RmtSendEfx

RmtRegisterUser

syntax :

RmtRegisterUser ***-user={<string$_{userId}$> <$string_{password}$>$_{opt}$}***

description :

Register a user on an Extended Fax FU. You must have a current authenticated session open with "Administrator" as the user and any password.

parameters:

Parameter	Description
-user	UserID / Password for session authentication

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtListAllUserId;
listRDataComplete(1)
Command OK (stuRmtListAllUserId)
> RmtRegisterUser -user={kqlee pass};
ACK
Command OK (stuRmtRegisterUser)
> RmtListAllUserId;
listRDataComplete(3)
userId=(kqlee)
Command OK (stuRmtListAllUserId)
```

RmtUnregisterUser

syntax :

RmtUnregisterUser *-userId*=<*string*> *-forceDelete*=<*bool*>

description :

Unregister a user on an Extended Fax FU. You must have a current authenticated session open with "Administrator" as the user and any password.

parameters:

Parameter	Description
-userId	User ID string
-forceDelete	Force delete of user and user-data even if user is not notified

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtListAllUserId;
listRDataComplete(7)
userId=(kqlee)
userId=(newuser)
Command OK (stuRmtListAllUserId)
> RmtUnregisterUser -userId=newuser -forceDelete=true;
ACK
Command OK (stuRmtUnregisterUser)
> RmtListAllUserId;
listRDataComplete(9)
userId=(kqlee)
Command OK (stuRmtListAllUserId)
```

RmtChangePassword

syntax :

RmtChangePassword ***-oldPassword=<string>*** *-newPassword=<string>*$_{opt}$

description :

Change a user password on an Extended Fax FU. The password is changed for the current session user only. The Administrator cannot change other user's password.

parameters:

Parameter	Description
-oldPassword	Password string that is currently registered with a Fax FU
-newPassword	New password string to be registered with a Fax FU

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

notes :

The original password is set with the RmtRegisterUser command. This command will change the password of the user of the current session only. The registered user password is not verified to the password entered when the authenticated session was opened.

example :

RmtListAllUserId

syntax :

RmtListAllUserId

description :

Retrieve a list of registered users on a Fax FU. You must have a current authenticated session open with “Administrator” as the user and any password.

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	RDATACOMPLETE	Data transfer from remote FU was completed, display decoded list
	RDATAABORT	Data transfer from remote FU was aborted
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtListAllUserId;
listRDataComplete(9)
userId=(kqlee)
Command OK (stuRmtListAllUserId)
```

RmtListEfxJob

syntax :

RmtListEfxJob

description :

Retrieve a list of fax jobs currently on the FU's fax job queue. Decode and display the list or indicate cause of failure.

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	RDATACOMPLETE	Data transfer from remote FU was completed, display decoded list
	RDATAABORT	Data transfer from remote FU was aborted
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtListEfxJob;
listRDataComplete(13)
jobHandle=(4096) requesterUserId=(Administrator) jobStatusCode=(1) issueTime=(11 August)
comment=(mycomment) dataSize=(80) numOfJobEntries=(2) requestPriority=(100)
jobHandle=(4097) requesterUserId=(Administrator) jobStatusCode=(1) issueTime=(11 August)
comment=(mycomment) dataSize=(80) numOfJobEntries=(3) requestPriority=(100)
jobHandle=(4098) requesterUserId=(Administrator) jobStatusCode=(1) issueTime=(11 August)
comment=(mycomment) dataSize=(40) numOfJobEntries=(2) requestPriority=(50)
Command OK (stuRmtListEfxJob)
```

RmtSubscribeEfxEvent

**** This command is not supported by the Client FU. The Efx device emulator will accept the command, but will not act upon it.*

syntax :

RmtSubscribeEfxEvent *-**genericSubscription**=<**enum**> -notifySlmFu={<ipaddress $_{slmIp}$> <int $_{fuHandle}$>}$_{opt}$ -checkInterval=<int>$_{opt}$ -targetUser={<string$_{userId}$> <string$_{password}$>}$_{opt}$*

description :

Subscribe to extended fax events.

parameters:

Parameter	Description
-genericSubscription	Specifies whether subscription is generic or not. Enumerated value - nonGeneric == (0), generic == (1)
-notifySlmFu	Address and FU ID to be notified
-checkInterval	Integer defining number of seconds between checks to see if remote FU is still active
-targetUser	UserID / Password for session authentication

result :

Command OK	ACK	Acknowledgement received, display subscription handle
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

notes :

You must create an authenticated session with userId="Administrator" prior to using this command.

example :

```
> RmtSubscribeEfxEvent -genericSubscription=1;
ACK
Arg1:int-value = 1
Command OK (stuRmtSubscribeEfxEvent)
```

RmtUnsubscribeEfxEvent

*** *This command is not supported by the Client FU. The Efx device emulator will accept the command, but will not act upon it.*

syntax :

RmtUnsubscribeEfxEvent ***-subscriptionHandle=<int>***

description :

Remove an existing extended fax event subscription.

parameters:

Parameter	Description
-subscriptionHandle	Dynamic status subscription handle given when the subscription was initiated.

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtUnsubscribeEfxEvent -subscriptionHandle=1;
ACK
Command OK (stuRmtUnsubscribeEfxEvent)
```

RmtRetrieveEfxData

**** This command is not supported by the Client FU. The Efx device emulator will accept the command, but will not act upon it.*

syntax :

RmtRetrieveEfxData *-dataId=<int>*

description :

Retrieve a data object from a remote extended fax FU.

parameters:

Parameter	Description
-dataId	ID of data that is to be retrieved.

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	RDATACOMPLETE	Data transfer from remote FU was completed
	RDATAABORT	Data transfer from remote FU was aborted
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

RmtRetrieveEfxDocId

*** *This command is not supported by the Client FU. The Efx device emulator will accept the command, but will not act upon it.*

syntax :

RmtRetrieveEfxDocId *-dataId=<int>*

description :

Retrieve an SLM, fuHandle, folderId, documentId, and FaxReadTime of a specific dataId which was stored on a (3rd party) docStore FU.

parameters:

Parameter	Description
-dataId	ID of data that is to be retrieved.

result :

Command OK	ACK	Acknowledgement received, display location parameters
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

RmtPrintEfxData

*** *This command is not supported by the Client FU. The Efx device emulator will accept the command, but will not act upon it.*

syntax :

RmtPrintEfxData *-dataId=<int>*

description :

Instruct Efx FU to print the specified dataId. Not every Efx FU can perform this command, so the user should first query the Fu's capability attributes to determine if this FU has the capability.

parameters:

Parameter	Description
-dataId	ID of data that is to be printed.

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

RmtQuerySentEfx

*** *This command is not supported by the Client FU. The Efx device emulator will accept the command, but will not act upon it.*

syntax :

RmtQuerySentEfx *-jobHandle=<int> -jobEntryId=( )intset*

description :

Send a querySentFax to get the SendExtFax related information. The command's job entry Id's must all belong to the same jobHandle.

parameters:

Parameter	Description
-jobHandle	Handle assigned to the job when it was received by the server.
-jobEntryId	Unique Id assigned to Job Entry

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	RDATACOMPLETE	Data transfer from remote FU was completed
	RDATAABORT	Data transfer from remote FU was aborted
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
	SDATAABORT	Data transfer to remote FU was aborted
Command ERROR		

example :

RmtQueryEfxHistory

*** *This command is not supported by the Client FU. The Efx device emulator will accept the command, but will not act upon it.*

syntax :

RmtQueryEfxHistory ***-userId=<string>***

description :

The User sends a *QueryFaxHistory* message to get all the history of Send Information, Receipt Information, and Read Information on the specified User at the Fax(C). An ordinary User can not query the fax history on the other Users. Only Administrator can QueryFaxHistory on the other Users.

parameters:

Parameter	Description
-userId	User ID fax history to query

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	RDATACOMPLETE	Data transfer from remote FU was completed
	RDATAABORT	Data transfer from remote FU was aborted
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
	SDATAABORT	Data transfer to remote FU was aborted
Command ERROR		

example :

RmtQueryReadInformation

syntax :

RmtQueryReadInformation ***-todHandle=<int>*** *-notificationMode={(<enumlist$_{jobStatusCode}$>) <bool$_{entries}$>}$_{opt}$ -notificationScheme={<ipaddress$_{slmIp}$><int$_{fuHandle}$>}$_{opt}$ -retryCount=<int>$_{opt}$ -retryInterval=<int>$_{opt}$ -scheduledDateTime=<string>$_{opt}$ -scheduledAfterTime=<string>$_{opt}$ -priority=<int>$_{opt}$*

description :

Send a queryReadInformation command to a remote FU. Any parameters specified in this command will override the defaults (if any) specified in the transfer data object. Many elements, especially those nested components, of the queryReadInformation definition must be built using todEfxQry commands prior to issuing a queryReadInformation.

parameters:

Parameter	Description
-todHandle	FU unique handle assigned to TOD when it is created
-notificationMode	Parameter includes a set of Job Status Codes and a boolean value allowing for a request for Job Entry Status. Only a Client FU can receive job status notifications. Enumerated list – see Salutation Architecture Specification, V2.0, Part 2, Page 201 for a listing of job status codes.
-notificationScheme	This parameter allows for notification to an FU other than the one that has initiated the job. Only a Client FU can receive job status notifications.
-retryCount	FU tries to re-call the remote telephone retryCount times, when the remote telephone was busy.
-retryInterval	FU tries to re-call the remote telephone at retryInterval minute intervals.
-scheduledDateTime	Time for scheduled future delivery. Expressed in UTCTime format.
-scheduledAfterTime	Time delay before delivery. Expressed in "hh:mm:ss" format.
-priority	Integer (0..100) -- 50 is the normal priority, -- 100 is the highest priority. Valid values are enumerated in the capability attributes on a per FU basis.

result :

Command OK	ACK	Acknowledgement received, display jobHandle
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

```
> RmtQueryReadInformation -todHandle=5 ;
Command OK (stuRmtQueryReadInformation)
```

RmtInformRead

**** This command is not supported by the Client FU. The Efx device emulator will accept the command, but will not act upon it.*

syntax :

RmtInformRead ***-dataId=<int> -hintDelete=<enum> -receiverInfo=<enum>***

description :

Inform Efx FU that the user has read the specified data.

parameters:

Parameter	Description
-dataId	ID of data that is to be reported upon reading.
-hintDelete	Shows that the User allows the [FaxData] FU to delete the Fax message. If the HintDelete is "delete", hereafter the User cannot find the Fax message on the [FaxData] FU. Enumerated value - delete == (0), do not delete == (1)
-receiverInfo	Enumerated value - others == (0), notRegisteredAtFax == (1), registeredAtFax == (2), registeredAtLocalSLM == (3), subscribing == (4), passedToGenericClient == (5)

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

example :

RmtSendEfx

syntax :

RmtSendEfx ***-todHandle=<int>*** *-dodHandle=<int>$_{opt}$ -modeOfDataTransfer=<enum>$_{opt}$ -dataSource={<ipaddress$_{slmIp}$><int$_{fuHandle}$>}$_{opt}$ -dataHandle=<int>$_{opt}$ -notificationMode={(<enumlist$_{jobStatusCode}$>) <bool$_{entries}$>}$_{opt}$ -notificationScheme={<ipaddress$_{slmIp}$><int$_{fuHandle}$>}$_{opt}$ -retryCount=<int>$_{opt}$ -retryInterval=<int>$_{opt}$ -queryInterval=<int>$_{opt}$ -coverSheetGeneration=<bool>$_{opt}$ -pageHeaderGeneration=<bool>$_{opt}$ -priority=<int>$_{opt}$*

description :

Send a sendEfx command to a remote FU. If client data is to be transferred, then a dodHandle, or a todHandle with associated data, must be specified. Any parameters specified in this command will override the defaults (if any) specified in the transfer data object. Many elements, especially nested components, of the sendEfx definition must be built using todEfx commands prior to issuing a sendExtFax.

parameters:

Parameter	Description
-todHandle	FU unique handle assigned to TOD when it is created. See caution in notes.
-dodHandle	FU unique handle assigned to DOD when it is created. Including a dodHandle will create an additional reference to that DOD. See caution in notes.
-modeOfDataTransfer	Enumerated list – immediate == 0, delayed == 1. If not defined, immediate is assumed.
-dataSource	Address of remote (3rd party) data source if data is not to be transferred from client FU. See caution in notes.
-dataHandle	Handle of remote (3rd party) data if data is not to be transferred from client FU. See caution in notes.
-notificationMode	Parameter includes a set of Job Status Codes and a boolean value allowing for a request for Job Entry Status. Only a Client FU can receive job status notifications. Enumerated list – see Salutation Architecture Specification, V2.0, Part 2, Page 201 for a listing of job status codes.
-notificationScheme	This parameter allows for notification to an FU other than the one that has initiated the job. Only a Client FU can receive job status notifications.
-retryCount	FU tries to re-call the remote telephone retryCount times, when the remote telephone was busy.
-retryInterval	Interval for FU retries to call the remote telephone (expressed in minutes)
-queryInterval	Value for when the client application needs to call to get Read Information. [FaxData] FU calls at “queryInterval” minute intervals.
-coverSheetGeneration	Value for whether a cover sheet will be generated. Boolean – include cover sheet == True, no cover sheet == False
-pageHeaderGeneration	Value for whether a page header will be added to the image. Boolean – include header == True, no header== False
-priority	Integer (0..100) -- 50 is the normal priority, -- 100 is the highest priority. Valid values are enumerated in the capability attributes on a per FU basis.

result :

Command OK	ACK	Acknowledgement received, display job Handle / entries
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
	SDATAABORT	Data transfer to remote FU was aborted
Command ERROR		

notes :

CAUTION: A valid data source must be defined for this command to function properly. There are three ways that the data source can be referenced:

1 – Reference a DOD with –dodHandle parameter.
2 – Reference a TOD with –todHandle parameter. The TOD must then reference a DOD.
3 – Reference a 3rd party data source with –dataSource and –dataHandle parameters.

example :

```
> RmtSendEfx –todHandle=4 –dodHandle=2 –notificationMode={(0 1 2 3) true} –retryCount=2;
jobHandle=4100
    jobEntryId=1
    jobEntryId=2
jobHandle=4101
    jobEntryId=1
    jobEntryId=2
    jobEntryId=3
Command OK (stuRmtSendEfx)
```

Unsupported Commands

FuShow

syntax :

FuShow

description :

Lists current status, ID's, size, and attributes of the FU

example :

```
> fushow;
FUNODE(1 ) name            = sfx
id              = 12000
ifu             = 0
hfu              = 1
jobQEnabled      = true
spoolStorageSize = 1048576
totalStorageSize = 1048576
 CAP_LIST (handles) :  10  11  20  30  40  41  42  50  51  60  12000  12001  12002  12003  12004
12005  12006  12010  12011  12012  12013  12020  12021  12030  12031  12032  12035  12036
12037  12038
 DATT_LIST (handles) :  996
 GATT_LIST (handles) :  12020  12021  12030  12031  12032  12035  12038
 MCB_LIST (handles) :  201  200  202  400  402  401  403  310  312  304  306  308  300  302  313
314  301  303  305  307  309  311  101  102  100  103  12000  12001  0  1
Command OK (stuFuShow)
```

TestSet

syntax :

TestSet *-id=<enum>*

description :

Configure the FU to generate a particular type of test or error condition. The test state will reset upon receipt of next command (the FU will return to normal).

parameters:

Parameter	Description
-id	0 - none 1 - timeout during next transaction. 2 - close session during next transaction 3 - return list queries with one response per segment 4 - issue a reverse session command during next transaction

Spy

syntax :

Spy *-id=<enum>*

description :

Perform an undocumented, developer specified function.

TraceOn

syntax :

TraceOn *-mask=<int>*$_{opt}$ *-fileName=<string>*$_{opt}$

description :

Start a system trace on the elements specified in the 'mask' parameter. Optionally store the trace to the specified filename.

TraceOff

syntax :

TraceOff

description :

Stop the system trace.

TodCheck

syntax :

TodCheck *-todHandle=<int>*

description :

Perform attribute checking on a TOD prior to queuing it locally (on job queue or document storage area of this FU). Verify range and bounds of job attributes associated with the TOD with respect to the capability attributes declared for the local FU.

parameters:

Parameter	Description
-todHandle	FU unique handle assigned to TOD when it is created.

result :

notes :

This command should NOT be performed prior to sending a job to a remote FU - since it verifies the TOD parameters with respect to the local FU (which is probably a client).

example :

```
TodCheck      -todHandle=<int>
```

DodWriteToFile

syntax :

DodWriteToFile *-parameter=<int>*

description :

description

parameters:

Parameter	Description
-xx	

result :

Command OK	ACK	Acknowledgement received
	NACK	Negative Acknowledgement Received
	XTIMEOUT	Transaction to remote FU timed out (no response)
	SVCCLOSE	Session to remote FU closed during transaction
Command ERROR		

notes :

Notes

example :

```
DodWriteToFile -parameter=<int>
```

SetVar

syntax :

SetVar *-name=<string> -value<string>*

description :

Sets a string value to a variable that can be used within a script. The variable can be used to replace any parameter type (integer, enumeration, boolean, or string).

To use the variable, prefix the variable name with $.

Use GetVar command to check the variable value.

parameters:

Parameter	Description
-name	variable name
-value	Value the variable will represent - If the value is numeric or includes white-space, then enclose the value in quotes.

result :

value set

example :

```
> sessionopen -slmip=127.0.0.1 -fuhandle=1;
svc handle = 4401636
Command OK (stuSessionOpen)
> SetVar -name="session1" -value="4401636";
> sessionselect -sessionhandle=$session1;
Command OK (stuSessionSelect)
```

GetVar

syntax :

GetVar *-name=<string>*

description :

Lists the value of a variable that has been previously set.

To use the variable, prefix the variable name with $.

Use SetVar command to set the variable value.

parameters:

Parameter	Description
-name	variable name assigned by SetVar command

result :

display variable value

example :

```
> SetVar -name="session1" -value="4401636";
> sessionselect -sessionhandle=$session1;
Command OK (stuSessionSelect)
> GetVar -name=session1;
$session1=4401636
```

Appendix

Job Status Code Table

Job Status Code	Job Status	Description
0	completed	-- job (entry) execution is successfully completed
1	queued	-- job (entry) execution has not begun yet
2	started	-- job (entry) execution has started and is being executed
3	suspended	-- job (entry) execution is suspended temporarily
4	resumed	-- suspended job (entry) execution is resumed
5	error	-- job (entry) execution is failed and was aborted
6	waitingForScheduledTime	-- job (entry) execution is scheduled at a later specific time
7	canceled	-- job (entry) was canceled by client's request
8	aborted	-- job (entry) execution was aborted by client's request

Attribute and Dynamic Status ID Tables

General

Common Capability Attributes

Capability Attribute Name	ID	Data Type	Compare Function ID
Major version	10	INTEGER	intEqualTo
Minor version	11	INTEGER	intGreaterThanOrEqualTo
Default coded character set	20	CharSetID	intEqualTo
FU name	30	DisplayString (SIZE(0..63))	strEqualTo
Manufacturer name	40	DisplayString (SIZE(0..63))	strEqualTo
Manufacturer product name	41	DisplayString (SIZE(0..63))	strEqualTo
Manufacturer product version	42	DisplayString (SIZE(0..63))	strEqualTo
Physical location	50	DisplayString (SIZE(0..255))	strEqualTo
Contact person name	51	DisplayString (SIZE(0..255))	strEqualTo
Authentication flavors	60	SET OF AuthenticationFlavor	setIntDoesContain

FU Default Capability Attribute Tables

Client FU

Attribute ID	Attribute Name	Attribute Type	Attribute Value(s)
10	Major version	int-value	2
11	Minor version	int-value	0
20	Default coded character set	int-value	17
30	FU name	octet-string-value	Granite
40	Manufacturer name	octet-string-value	IBM
41	Manufacturer product name	octet-string-value	test tool
42	Manufacturer product version	octet-string-value	Ver1.0
50	Physical location	octet-string-value	117 9th Av , Parker CO 80687
51	Contact person name	octet-string-value	contact@salutation.org
60	Authentication flavors	set-value(s)	0 1
1000	User ID	octet-string-value	kqlee

Print FU

Attribute ID	Attribute Name	Attribute Type	Attribute Value(s)
10	Major version	int-value	2
11	Minor version	int-value	0
20	Default coded character set	int-value	17
30	FU name	octet-string-value	Granite
40	Manufacturer name	octet-string-value	IBM
41	Manufacturer product name	octet-string-value	Jake
42	Manufacturer product version	octet-string-value	Ver1.0
50	Physical location	octet-string-value	117 9th Av , Parker CO 80687
51	Contact person name	octet-string-value	contact@salutation.org
60	Authentication flavors	set-value(s)	0 1
10000	personalityProtocol	set-value(s)	1203 1205 1206
10001	supportedCommand	set-value(s)	1203 1205 1206
10002	dynamicStatusId	set-value(s)	10000 10004
10003	spoolStorage	bool-value	TRUE
10004	minimumCheckInterval	bool-value	FALSE
10010	documentFormat	set-value(s)	1203 1205 1206
10011	imageCompAlgorithm	set-value(s)	0
10012	imageByteFillOrder	set-value(s)	0
10013	imageResolution	set-value(s)	0
10020	printPaperSize	set-value(s)	1 21
10021	printResolution	set-value(s)	0 10
10022	printPaperDirection	set-value(s)	1 2
10023	printCopyCount	int-value	50
10024	printPaperInputSelect	set-value(s)	126 1
10025	printPaperOutputSelect	set-value(s)	0
10026	printOutputBinSelect	int-value	6
10027	printDuplexMode	set-value(s)	0 1
10028	maximumBindingMargin	int-value	10
10029	printFaceUpMode	set-value(s)	1
10030	printStaplingSelect	set-value(s)	0
10040	printPriority	set-value(s)	100 50 0
10041	modeOfDataTransfer	set-value(s)	0 1
10042	dataLocationScheme	set-value(s)	0
10043	dataTransferTimeOutSettable	bool-value	TRUE
10044	dataTransferTimeOutLength	int-value	120

Doc FU

Attribute ID	Attribute Name	Attribute Type	Attribute Value(s)
10	Major version	int-value	2
11	Minor version	int-value	0
20	Default coded character set	int-value	17
30	FU name	octet-string-value	Granite
40	Manufacturer name	octet-string-value	IBM
41	Manufacturer product name	octet-string-value	Jake
42	Manufacturer product version	octet-string-value	Ver1.0
50	Physical location	octet-string-value	117 9th Av , Parker CO 80687
51	Contact person name	octet-string-value	contact@salutation.org
60	Authentication flavors	set-value(s)	0 1
1000	User ID	octet-string-value	kqlee
11000	personalityProtocol	set-value(s)	1
11001	supportedCommand	set-value(s)	1 30
11002	dynamicStatusId	set-value(s)	11000 11001 11002
11003	readWriteCapability	set-value(s)	2
11004	minimumCheckInterval	int-value	0
11010	modeOfStore	set-value(s)	1 2
11011	documentFormat	set-value(s)	127
11012	imageCompAlgorithm	set-value(s)	127
11013	imageByteFillOrder	set-value(s)	0
11014	imageResolution	set-value(s)	0
11030	dataLocationScheme	set-value(s)	0
11031	dataTransferTimeOutSettable	bool-value	TRUE
11032	dataTransferTimeOutLength	int-value	120

Send Fax FU

Attribute ID	Attribute Name	Attribute Type	Attribute Value(s)
10	Major version	int-value	2
11	Minor version	int-value	0
20	Default coded character set	int-value	17
30	FU name	octet-string-value	Granite
40	Manufacturer name	octet-string-value	IBM
41	Manufacturer product name	octet-string-value	Jake
42	Manufacturer product version	octet-string-value	Ver1.0
50	Physical location	octet-string-value	117 9th Av , Parker CO 80687
51	Contact person name	octet-string-value	contact@salutation.org
60	Authentication flavors	set-value(s)	0 1
12000	personalityProtocol	set-value(s)	1
12001	supportedCommand	set-value(s)	1 2 3
12002	dynamicStatusId	set-value(s)	12000 12001 12002
12003	numOfCalledSubscribers	int-value	100
12004	spoolStorage	bool-value	TRUE
12005	faxSendOrdering	bool-value	TRUE
12006	minimumCheckInterval	int-value	0
12010	documentFormat	set-value(s)	127
12011	imageCompAlgorithm	set-value(s)	127
12012	imageByteFillOrder	set-value(s)	0
12013	imageResolution	set-value(s)	0
12020	coverSheetGen	bool-value	TRUE
12021	pageHeaderGen	bool-value	TRUE
12030	faxProtocol	int-value	6038664
12031	requestPriority	set-value(s)	100 50 0
12032	retryCount	int-value	100
12035	modeOfDataTransfer	set-value(s)	0
12036	dataLocationScheme	set-value(s)	0
12037	dataTransferTimeOutSettable	bool-value	TRUE
12038	dataTransferTimeOutLength	int-value	120

Extended Fax FU

Attribute ID	Attribute Name	Attribute Type	Attribute Value(s)
10	Major version	int-value	2
11	Minor version	int-value	0
20	Default coded character set	int-value	17
30	FU name	octet-string-value	Granite
40	Manufacturer name	octet-string-value	IBM
41	Manufacturer product name	octet-string-value	Jake
42	Manufacturer product version	octet-string-value	Ver1.0
50	Physical location	octet-string-value	117 9th Av , Parker CO 80687
51	Contact person name	octet-string-value	contact@salutation.org
60	Authentication flavors	set-value(s)	0 1
13000	personalityProtocol	set-value(s)	1203 1205 1206
13001	supportedCommand	set-value(s)	1203 1205 1206
13002	dynamicStatusId	set-value(s)	10000 10004
13003	numOfFaxEventSubscribers	int-value	100
13004	faxSendOrdering	bool-value	FALSE
13005	minimumCheckInterval	int-value	0
13010	documentFormat	set-value(s)	1203 1205 1206
13011	imageCompAlgorithm	set-value(s)	0
13012	imageByteFillOrder	set-value(s)	0
13013	imageResolution	set-value(s)	0
13020	retryCount	int-value	100
13021	queryInterval	int-value	100
13022	timeOutReadConfirmation	int-value	100
13023	retryInterval	int-value	100
13024	modeOfDataTransfer	set-value(s)	0
13025	faxExtensionCapability	set-value(s)	0
13026	maxUserID	int-value	100
13027	maxNumberOfClientPerUser	int-value	100
13031	dataTransferTimeOutSettable	bool-value	TRUE
13032	dataTransferTimeOutLength	int-value	120
13033	storageAlarmLevel	int-value	100
13034	requestPriority	set-value(s)	100 50 0

Test Scripts

The following scripts will be useful to the user for examples of command usage. These scripts were written as a method for quickly performing regression testing of the commands included in the Salutation test tool.

The scripts assume that the script files are stored in a directory called “scripts”. This directory must be a subdirectory of the “bin” directory that holds the test tool files.

Command Type	**Script Name**
SLM	
Session	
TOD	Tod.txt
DOD	Dod.txt
Global Attributes	Attr.txt
Private Attributes	
Rmt Attributes	
Job	Job.txt
Job Entry	JobEntry.txt
Rmt Job	JobRmt.txt
Rmt JobEnt	JobEntryRmt.txt
Print FU	Print.txt
Rmt Print	
Doc FU	Doc.txt
Rmt Doc	DocRmt.txt
Rmt Data Transfer	
Sfx FU	Sfx.txt
Rmt Sfx	SfxRmt.txt
Efx FU	Efx.txt
Rmt Efx	EfxRmt.txt
Rmt Dynamic Status	

Tod Script

```
#########################################################################
#  Tod Script
#  Tod.txt
#  Granite Systems
#  D. Seger  14-Nov-99
#########################################################################

# This script is designed to test the following Tod Commands
#       TodDestroy
#       TodStoreDocCreate
#       TodPrintCreate
#       TodSendSfxCreate
#       TodSendSfxTsinfoSet
#       TodSendSfxCsinfoPut
#       TodQueryReadInformationCreate
#       TodQueryReadInformationEfqPut
#       TodQueryReadInformationEfqEntryPut
#       TodSendEfxCreate
#       TodSendEfxSuiSet
#       TodSendEfxFsiSet
#       TodSendEfxSdiPut
#       TodSendEfxSdiRcoiSet
#       TodSendEfxSdiEntryPut
#       TodDodSet
#       TodListShow
#       TodShow

##  Run this script as -->  script -scr=scripts\tod.txt -errorContinue=true

##  Manual set-up
        # Create an FU of any type
        # Run this script on the FU

##  Script set-up
        DodCreateFromChars -dodHandle=1 -chr=63 -chcount=20 -segcount=2 -blockcount=2;
        DodCreateFromChars -dodHandle=2 -chr=63 -chcount=10  -segcount=4 -blockcount=1;
        echo -str="List SLM's on Network Segment";
        SlmListFU -slmIp=127.0.0.1;


TodListShow;                                  # should return null list

##########  Print Tod Commands  #############

TodPrintCreate
    -todHandle=1                                        # this TOD is later destroyed
    -dodHandle=1
    -modeOfDataTransfer=0                   # immediate
#    -dataSource={ 127.0.0.1 1 }
#    -dataHandle=1
    -life=0                                 # job
#    -notificationMode={(1 2 3 4) false}
#    -notificationScheme={ 127.0.0.1 1 }
    -paperSize=1                            # paper size letter
    -resolution=0                           # image resolution normal
    -paperDirection=1                       # portrait
    -copyCount=1
    -inputSelect=126                        # auto
    -outputSelect=0                         # standard
    -outputBinSelect=1
    -duplexModeSelect=0                     # simplex
    -duplexBindingMargin=0
    -faceUpModeSelect=1                     # face down
    -priority=50                            # normal
    -staplingSelect=0                       # none
    -fileName=printFile;

TodPrintCreate
    -todHandle=6
    -dodHandle=1
    -modeOfDataTransfer=0                   # immediate
#    -dataSource={ 127.0.0.1 1 }
```

```
#    -dataHandle=1
    -life=0                                 # job
#    -notificationMode={(1 2 3 4) false}
#    -notificationScheme={ 127.0.0.1 1 }
    -paperSize=1                            # paper size letter
    -resolution=0                           # image resolution normal
    -paperDirection=1                       # portrait
    -copyCount=1
    -inputSelect=126                        # auto
    -outputSelect=0                         # standard
    -outputBinSelect=1
    -duplexModeSelect=0                     # simplex
    -duplexBindingMargin=0
    -faceUpModeSelect=1                     # face down
    -priority=50                            # normal
    -staplingSelect=0                       # none
    -fileName=printFile;

########## StoreDoc Tod Commands  #############

TodStoreDocCreate
    -todHandle=2
    -dodHandle=1                                        # later set to dodHandle=2
    -folderId=1
#   -dataSource={ 127.0.0.1 1 }
#   -dataHandle=1
    -modeOfStore=1                          # document data mode
    -ownerName="doc owner"
    -docComment="doc comment";


##########  Sfx Tod Commands  #############

TodSendSfxCreate
    -todHandle=3
    -dodHandle=1
    -modeOfDataTransfer=0                   # immediate
#   -dataSource={ 127.0.0.1 1 }
#   -dataHandle=1
    -life=0                                 # job
    -notificationMode={(1 2 3 4) false}
#   -notificationScheme={ 127.0.0.1 1 }
    -priority=50                            # normal
    -retryCount=1;

TodSendSfxTsinfoSet
    -todHandle=3
    -name=kerry
    -company=granite
    -phoneNumber="303-555-1122"
    -faxNumber="303-555-2233"
    -address="123 Main Street, Longmont , CO"
    -subject="sfx fax"
# !!    -coverSheetGen=true
    -memoForCover="cover memo"
# !!    -pageHeaderGen=true
    -memoForHeader="header memo";

TodSendSfxCsinfoPut
    -todHandle=3
    -jobEntryId=1
    -faxNumber="303-555-3344"
    -subAddressNumber="2000"
    -name="Mike Osborn"
    -section="Business Management"
    -company="Granite Systems"
    -phoneNumber="303-555-4455"
    -address="234 Main Street , Boulder , CO"
    -faxProtocol=4                                  # !! should be enum
    -orderingData="myorderingdata";

TodSendSfxCsinfoPut
    -todHandle=3
    -jobEntryId=2
    -faxNumber="303-555-4433"
    -subAddressNumber="3000"
```

```
    -name="Dennis Seger"
    -section="Technical Writer"
    -company="Granite Systems"
    -phoneNumber="303-555-5544"
    -address="432 Main Street , Boulder , CO"
    -faxProtocol=4                              # !! should be enum
    -orderingData="yourorderingdata";

##########  Efx Tod Commands  #############

TodSendEfxCreate
    -todHandle=4
    -dodHandle=1
    -retryCount=4
    -retryInterval=1
    -queryInterval=1
    -coverSheetGeneration=true
    -pageHeaderGeneration=true
    -priority=100;                          # high

TodSendEfxSuiSet
    -todHandle=4
    -senderUserId=kqlee
    -sourceFaxNumber="303-555-5566"
    -name=kerry
    -section=networks
    -company=granite
    -phoneNumber="303-555-6677"
    -address="345 Main Street , Longmont, CO";

TodSendEfxFsiSet
    -todHandle=4
    -issueTime="11 August"
    -comment="mycomment";

TodSendEfxSdiPut
    -todHandle=4
    -sdiHandle=1
    -destinationFaxNumber="303-555-7788";

TodSendEfxSdiRcoiSet
    -todHandle=4
    -sdiHandle=1
    -returnMethod=0
    -timeOut=60
    -callDirectionOrder=2;

TodSendEfxSdiEntryPut
    -todHandle=4
    -sdiHandle=1
    -jobEntryId=1
    -receiverUserID=user410
    -name=kqlee410
    -section=section410
    -company=company410
    -phoneNumber="303-555-8899";


TodSendEfxSdiEntryPut
    -todHandle=4
    -sdiHandle=1
    -jobEntryId=2
    -receiverUserID=user412a
    -name=kqlee412
    -section=section412a
    -company=company412a
    -phoneNumber="303-555-9900";

TodSendEfxSdiPut
    -todHandle=4
    -sdiHandle=2
    -destinationFaxNumber="303-555-0011";

TodSendEfxSdiRcoiSet
    -todHandle=4
    -sdiHandle=2
```

```
    -returnMethod=1
    -timeOut=30
    -callDirectionOrder=0;

TodSendEfxSdiEntryPut
    -todHandle=4
    -sdiHandle=2
    -jobEntryId=1
    -receiverUserID=user421b
    -name=kqlee421b
    -section=section421b
    -company=company421b
    -phoneNumber="303-555-1234";

TodSendEfxSdiEntryPut
    -todHandle=4
    -sdiHandle=2
    -jobEntryId=2
    -receiverUserID=user422
    -name=kqlee422
    -section=section422
    -company=company422
    -phoneNumber="303-555-2345";

TodSendEfxSdiEntryPut
    -todHandle=4
    -sdiHandle=2
    -jobEntryId=3
    -receiverUserID=user423
    -name=kqlee423
    -section=section423
    -company=company423
    -phoneNumber="303-555-3456";

TodQueryReadInformationCreate
    -todHandle=5
    -notificationMode={ (1 2 4) false }
    -notificationScheme={ 127.0.0.1 1 }
    -retryCount=5
    -retryInterval=5
    -scheduledDateTime="12:38 11 Oct 1999"
    -scheduledAfterTime="13:38 11 Oct 1999"
    -priority=100;

TodQueryReadInformationEfqPut
    -todHandle=5
    -sourceJobhandle=4098;

TodQueryReadInformationEfqEntryPut
    -todHandle=5
    -sourceJobHandle=4098
    -jobEntryId=1
    -userId=kqlee501;

TodQueryReadInformationEfqEntryPut
    -todHandle=5
    -sourceJobHandle=4098
    -jobEntryId=2
    -userId=kqlee502;

##########  Misc. Tod Commands  #############

TodDestroy
    -todHandle=1;

TodDodSet
    -todHandle=2
    -dodHandle=2;

TodListShow;

TodShow
    -todHandle=4;

Echo -str="   ### END OF SCRIPT ###";
```

Dod Script

```
#########################################################################
#  Dod Script
#  Dod.txt
#  Granite Systems
#  D. Seger  08-Dec-99
#########################################################################

# This script is designed to test the following DOD Commands
#       DodCreateFromChars
#       DodCreateFromFile
#       DodDestroy
#       DodListShow
#       DodShow

##  Run this script as -->  script -scr=scripts\dod.txt -errorContinue=true

##  Manual set-up
        # Create a DocStore FU
        # Create a Client FU with storage of 1 Mb.
        # Open a session from the Client FU to the DocStore FU
        # Run this script on the Client FU

##  Script set-up
        RmtCreateFolder;
        RmtCreateFolder;


##########  Create and List DOD's  ############

DodCreateFromChars                  #  Create a DOD from characters
  -dodHandle=100
  -chr=30
  -chCount=100
  -segCount=1
  -blockCount=1;

DodCreateFromChars                  #  Create a second DOD from characters
  -dodHandle=101
  -chr=45
  -chCount=1000
  -segCount=4
  -blockCount=3;

DodCreateFromChars                  #  Attempt to create a DOD larger than the spool size
  -dodHandle=102
  -chr=30
  -chCount=10000
  -segCount=1000
  -blockCount=1000;

Echo -str="   ^^^ should have failed - not enough storage";

DodCreateFromChars                  #  Attempt to create a DOD using a duplicate DOD Handle
  -dodHandle=101
  -chr=45
  -chCount=25
  -segCount=100
  -blockCount=1;

Echo -str="   ^^^ should have failed - dodHandle already exists";

DodCreateFromFile                           #  Create a DOD from a file
  -dodHandle=103
  -filename="scripts\tod.txt";

DodCreateFromFile                           #  Create a second DOD from a file
  -dodHandle=104
  -filename="scripts\print.txt";

DodCreateFromFile                           #  Attempt to create a DOD from a non-existent file
  -dodHandle=105
  -filename="nofilebythisname.123";
```

```
Echo -str="   ^^^ should have failed - could not open file";

DodCreateFromFile                    #  Attempt to create a DOD from a file larger than the
spool size
   -dodHandle=106
   -filename="fumaster.exe";

        Echo -str="   ^^^ should have failed - not enough storage";

DodShow                              #  Listing of dodHandle=100
  -dodHandle=100;

Echo -str="   ^^^ should list details of DOD 100 ";

DodListShow;                         #  Listing of all DOD's

Echo -str="   ^^^ should list only DOD's 100, 101, 103, 104 ";



##########  Transfer the DOD's to a Remote FU  ############

RmtStoreDoc                                  #  Transfer a single seg & block DOD to a remote
DocStore
  -dodHandle=100                             #  Created from characters
  -folderId=0;                       #  Will be docHandle=1

RmtStoreDoc                                  #  Transfer a multiple seg & block DOD to a remote
DocStore
  -dodHandle=101                             #  Created from characters
  -folderId=0;                               #  Will be docHandle=2

RmtStoreDoc                                  #  Transfer a DOD to a remote DocStore
  -dodHandle=103                             #  Created from file
  -folderId=0                        #  Will be docHandle=3
  -modeOfStore=1;                            #  keep as 3 blocks

RmtStoreDoc                                  #  Do it again so there are two references to the
DOD 103
  -dodHandle=103
  -folderId=1;                               #  Will be docHandle=4

RmtListFolderDoc                     #  List doc in folder
  -folderId=0;

Echo -str="   ^^^ should list Doc's 1, 2, 3 ";

RmtListFolderDoc                     #  List doc in folder
  -folderId=1;

Echo -str="   ^^^ should list Doc's 4 ";


##########  Destroy and List DOD's  ############

DodDestroy                                   #  Delete DOD 104
  -dodHandle=104;

DodListShow;                         #  Listing of all DOD's

        Echo -str="   ^^^ should list only DOD's 100, 101, 103 ";

DodShow                              #  Attempt to list destroyed dodHandle=104
  -dodHandle=104;

        Echo -str="   ^^^ should fail - invalid dod handle";

DodShow                              #  List DOD 103 details
  -dodHandle=103;

        Echo -str="   ^^^ should show dataHandle=103";

RmtDeleteDoc                         #  Delete one of the documents ref'ed by DOD 103
  -folderId=1                        #  This should cause other reference to be destroyed
  -documentId=4;
```

```
DodListShow;                          #  Listing of all DOD's

Echo -str="   ^^^ should list DOD's 100, 101, 103 ";


Echo -str="   ### END OF SCRIPT ###";
```

Attributes Script

```
#########################################################################
#  Attributes Script
#  Attr.txt
#  Granite Systems
#  D. Seger  08-Dec-99
#########################################################################

# This script is designed to test the following Attribute Commands
#       GlobalAttributeSet
#       GlobalAttributeGet
#       GlobalAttributeListShow
#       PrivateAttributeSet
#       PrivateAttributeGet
#       PrivateAttributeListShow
#       RmtGetGlobalAttribute
#       RmtGetPrivateAttribute
#       RmtSetPrivateAttribute

##  Run this script as -->  script -scr=scripts\attr.txt -errorContinue=true

##  Manual set-up
        # Create a DocStore FU (#1)
        # Create another DocStore FU (#2)
        # Open a session from #1 to #2
        # Set the DocStore #1 sessionHandle to a variable --> setvar -name="sh1" -
value="<sessionHandle>"

        # Create a SendFax FU (#3)
        # Open a session from #1 to #3
        # Set the DocStore #1 sessionHandle to a variable --> setvar -name="sh2" -
value="<sessionHandle>"

        # Run the script on the DocStore FU #1


##  Script set-up
        getvar -name=sh1;                    # verify variable value
        getvar -name=sh2;                    # verify variable value
        SessionSelect -sessionHandle=$sh1;


##########  Global Attribute Commands  ############

GlobalAttributeSetInt
  -attributeId=11001                # not a global attribute, but command doesn't care
  -attributeValue=123;

GlobalAttributeGet
  -attributeId=11001;

        Echo -str="   ^^^   Should list int-value = 123 ";

GlobalAttributeSetEnum
  -attributeId=11002                # not a global attribute, but command doesn't care
  -attributeValue=2;

GlobalAttributeGet
  -attributeId=11002;

        Echo -str="   ^^^   Should list enum-value = 2 ";

GlobalAttributeSetBool
  -attributeId=11003                # not a global attribute, but command doesn't care
  -attributeValue=true;

GlobalAttributeGet
  -attributeId=11003;

        Echo -str="   ^^^   Should list bool-value = TRUE ";

GlobalAttributeSetString
  -attributeId=11004                # not a global attribute, but command doesn't care
```

```
  -attributeValue="Granite Systems";

GlobalAttributeGet
  -attributeId=11004;

        Echo -str="   ^^^   Should list octet-string-value = Granite Systems ";

GlobalAttributeListShow;


##########  Private Attribute Commands  ############

PrivateAttributeSetInt
  -sessionHandle=$sh1
  -attributeId=11001
  -attributeValue=456;

PrivateAttributeGet
  -sessionHandle=$sh1
  -attributeId=11001;

        Echo -str="   ^^^   Should list int-value = 456 ";

PrivateAttributeSetEnum
  -sessionHandle=$sh1
  -attributeId=11002
  -attributeValue=4;

PrivateAttributeGet
  -sessionHandle=$sh1
  -attributeId=11002;

        Echo -str="   ^^^   Should list enum-value = 4 ";

PrivateAttributeSetBool
  -sessionHandle=$sh1
  -attributeId=11003
  -attributeValue=False;

PrivateAttributeGet
  -sessionHandle=$sh1
  -attributeId=11003;

        Echo -str="   ^^^   Should list bool-value = FALSE ";

PrivateAttributeSetString
  -sessionHandle=$sh1
  -attributeId=11004
  -attributeValue=IBM;

PrivateAttributeGet
  -sessionHandle=$sh1
  -attributeId=11004;

        Echo -str="   ^^^   Should list octet-string-value = IBM ";

PrivateAttributeListShow
  -sessionHandle=$sh1;




##########  Remote Attribute Commands  ############

SessionSelect -sessionHandle=$sh2;

RmtGetGlobalAttribute
  -attributeId=(12021 12099);        # 12021=bool 12099=invalid

        Echo -str="   ^^^   Should list attributeId = 12021 bool-value = true ";

RmtGetGlobalAttribute
  -attributeId=(12031 12032);        # 12031=enum 12032=integer

        Echo -str="   ^^^   Should list attributeId = 12031 enum-value = 50 12032 int-value = 100
";
```

```
SessionSelect -sessionHandle=$sh1;


RmtGetPrivateAttribute
  -attributeid=(11010);

       Echo -str="   ^^^   Should list No Attributes ";

RmtSetPrivateAttributeEnum
  -attributeId=11010
  -attributeValue=1;

RmtGetPrivateAttribute
  -attributeid=(11010);

       Echo -str="   ^^^   Should list attributeId = 11010 enum-value = 1 ";

RmtSetPrivateAttributeEnum                  # clear private attribute
  -attributeId=11010;

RmtGetPrivateAttribute
  -attributeid=(11010);

       Echo -str="   ^^^   Should list No Attributes ";

RmtGetPrivateAttribute
  -attributeid=(12099);                     # invalid attribute ID

       Echo -str="   ^^^   Should list No Attributes ";

##########  Invalid Attribute Commands  ############
       Echo -str="           ^^^  all commands after this point should fail  ^^^";

GlobalAttributeGet
  -attributeId=99;                  # invalid attribute ID

PrivateAttributeGet
  -sessionHandle=$sh1
  -attributeId=99;                  # invalid attribute ID

PrivateAttributeGet
  -sessionHandle=99                 # invalid session handle
  -attributeId=11004;

PrivateAttributeListShow
  -sessionhandle=99;                # invalid session handle

SessionSelect -sessionHandle=$sh2;

RmtSetPrivateAttributeInt           # wrong type for 12021
  -attributeId=12021
  -attributeValue=99;

Echo -str="   ### END OF SCRIPT ###";
```

Job Local Script

```
########################################################################
#  Job Local Script
#  Job.txt
#  Granite Systems
#  D. Seger  14-Nov-99
########################################################################

# This script is designed to test the following Job Commands
#       JobStart
#       JobComplete
#       JobCancel
#       JobAbort
#       JobError
#       JobAttributeGet
#       JobAttributeSet
#       JobListShow
#       JobShow

##  Run this script as -->  script -scr=scripts\job.txt -errorContinue=true

##  Manual set-up
        # Create a Print FU
        # Run this script on the Print FU

##  Script set-up
        DodCreateFromChars -dodHandle=100 -chr=30 -chCount=100 -segCount=1-blockCount=1;
        PrnPrint -dodhandle=100 -life=0;             # will have jobHandle=4096
        PrnPrint -dodhandle=100 -life=1;             # will have jobHandle=4097
        PrnPrint -dodhandle=100 -life=2;             # will have jobHandle=4098


##########  Job Queue Commands  ############

JobQEnable;

PrnPrint                                      # will have jobHandle=4099
  -dodhandle=100
  -life=0;


##########  Job Status Commands  ############

JobShow
  -jobHandle=4096;

        Echo -str="   ^^^  Should have job status code = 1";

JobStart
  -jobHandle=4096;

JobShow
  -jobHandle=4096;

        Echo -str="   ^^^  Should have job status code = 2";

JobComplete
  -jobHandle=4096;

JobListShow;

        Echo -str="   ^^^  List should have 4097 4098 4099";

JobCancel
  -jobHandle=4097;

JobShow
  -jobHandle=4097;

        Echo -str="   ^^^  Should have job status code = 7";

JobAbort
  -jobHandle=4098;
```

```
JobShow
  -jobHandle=4098;

        Echo -str="   ^^^  Should have job status code = 8";

JobError
  -jobHandle=4099;

JobListShow;

        Echo -str="   ^^^  List should have 4097 4098";


##########  Job Attribute Commands  #############

JobAttributeGet                              # get current job priority
  -jobHandle=4097
  -attributeId=10040;

JobAttributeSetEnum                  # set job priority to 75
  -jobHandle=4097
  -attributeId=10040
  -attributeValue=75;

JobAttributeGet                              # verify new job priority
  -jobHandle=4097
  -attributeId=10040;

JobAttributeSetInt                   # set job priority to 80 (even though it's an Enum)
  -jobHandle=4097
  -attributeId=10040
  -attributeValue=80;

JobAttributeGet                              # verify new job priority
  -jobHandle=4097
  -attributeId=10040;

JobAttributeSetBool                  # set job priority to True (even though it's an Enum)
  -jobHandle=4097
  -attributeId=10040
  -attributeValue=True;

JobAttributeGet                              # verify new job priority
  -jobHandle=4097
  -attributeId=10040;

JobAttributeSetString                # set job priority to "Granite" (even though it's an Enum)
  -jobHandle=4097
  -attributeId=10040
  -attributeValue="Granite";

JobAttributeGet                              # verify new job priority
  -jobHandle=4097
  -attributeId=10040;


##########  Invalid Job Commands  #############
        Echo -str="           ^^^  all commands after this point should fail  ^^^";

JobStart
  -jobHandle=4096;                   # invalid job handle

JobComplete
  -jobHandle=4096;                   # invalid job handle

JobCancel
  -jobHandle=4096;                   # invalid job handle

JobError
  -jobHandle=4096;                   # invalid job handle

JobAbort
  -jobHandle=4096;                   # invalid job handle

JobShow
```

```
  -jobHandle=4096;                  # invalid job handle

JobAttributeSetEnum
  -jobHandle=4096                          # invalid job handle
  -attributeId=10040
  -attributeValue=75;

JobAttributeGet
  -jobHandle=4096                          # invalid job handle
  -attributeId=10040;

JobAttributeGet
  -jobHandle=4097
  -attributeId=10444;               # invalid attribute ID



Echo -str="   ### END OF SCRIPT ###";
```

Job Remote Script

```
#########################################################################
#  Job Remote Script
#  JobRmt.txt
#  Granite Systems
#  D. Seger  08-Dec-99
#########################################################################

# This script is designed to test the following Job Commands
#       RmtQueryJobStatus
#       RmtCancelJob
#       RmtSuspendJob
#       RmtResumeJob
#       RmtChangeJobAttribute
#       RmtStartMonitorJobStatus
#       RmtCancelMonitorJobStatus
#       RmtFreeJobHandle

##  Run this script as -->  script -scr=scripts\jobrmt.txt -errorContinue=true

##  Manual set-up
        # Create a Print FU
        # Create a Client FU
        # Open a session from the Client to the Print FU
        # Run this script on the Client FU


##  Script set-up
        DodCreateFromChars -dodHandle=100 -chr=30 -chCount=100 -segCount=1-blockCount=1;
        RmtPrint -dodhandle=100 -life=0;             # will have jobHandle=4096
        RmtPrint -dodhandle=100 -life=2;             # will have jobHandle=4097
        RmtPrint -dodhandle=100 -life=1;             # will have jobHandle=4098
        RmtPrint -dodhandle=100 -life=2;             # will have jobHandle=4099



##########  Remote Job Status Commands  ############

RmtStartMonitorJobStatus                    # subscribe to all status's
  -jobHandle=4096
  -notificationMode={(0 1 2 3 4 5 6 7 8) true};

RmtStartMonitorJobStatus                    # subscribe to only Cancel status
  -jobHandle=4097
  -notificationMode={(7) true};

RmtQueryJobStatus
  -jobHandle=4096;

        Echo -str="   ^^^  Should be job status = 1";

RmtSuspendJob                               # suspend job
  -jobHandle=4096;

        Echo -str="   ^^^  Should be notification of job 4096 = 3";

RmtQueryJobStatus
  -jobHandle=4096;

        Echo -str="   ^^^  Should be job status = 3";

RmtCancelJob                                # cancel job - it should be deleted since life=0
  -jobHandle=4096
  -abort=true;

        Echo -str="   ^^^  Should be notification of job 4096 = 7";

RmtFreeJobHandle
  -jobHandle=4097;

        Echo -str="   ^^^  Should fail for illegal state transition";

RmtCancelJob                                # cancel job; it should persist since life=2
```

```
  -jobHandle=4097
  -abort=false;

        Echo -str="   ^^^  Should be notification of job 4097 = 7";

RmtQueryJobStatus
  -jobHandle=4097;

        Echo -str="   ^^^  Should be job status = 7";

RmtStartMonitorJobStatus                    # subscribe to Suspend and Resume status's
  -jobHandle=4098
  -notificationMode={(3 4) false};

RmtSuspendJob
  -jobHandle=4098;

RmtCancelMonitorJobStatus
  -jobhandle=4098;

RmtResumeJob
  -jobHandle=4098;

        Echo -str="   ^^^  Should be notification of job 4098 = 3 and not 4098 = 4";

RmtQueryJobStatus
  -jobHandle=4098;

        Echo -str="   ^^^  Should be job status = 1";

RmtFreeJobHandle
  -jobHandle=4097;


##########  Job Attribute Commands  #############

RmtChangeJobAttributeEnum                   # set job priority to 50
  -jobHandle=4098
  -attributeId=10040
  -attributeValue=50;


##########  Invalid Job Commands  #############
        Echo -str="           ^^^  all commands after this point should fail  ^^^";

RmtStartMonitorJobStatus
  -jobHandle=4999                                  # invalid job handle
  -notificationMode={(7) true};

RmtCancelMonitorJobStatus
  -jobhandle=4999;                          # invalid job handle

RmtQueryJobStatus
  -jobHandle=4096;                          # invalid job handle - job was cancelled

RmtQueryJobStatus
  -jobHandle=4999;                          # invalid job handle

RmtQueryJobStatus
  -jobHandle=4097;                          # invalid job handle - job handle was freed

RmtSuspendJob
  -jobHandle=4999;                          # invalid job handle

RmtCancelJob
  -jobHandle=4999                           # invalid job handle
  -abort=true;

RmtResumeJob
  -jobHandle=4999;                          # invalid job handle

RmtFreeJobHandle
  -jobHandle=4999;                          # invalid job handle

RmtFreeJobHandle
  -jobHandle=4099;                          # Cannot free a job handle if job is queued
```

```
RmtChangeJobAttributeInt                    # try to set an Enumeration as an Integer
  -jobHandle=4098
  -attributeId=10040
  -attributeValue=80;

RmtChangeJobAttributeInt
  -jobHandle=99                                     # invalid job handle
  -attributeId=10040
  -attributeValue=80;

Echo -str="   ### END OF SCRIPT ###";
```

Job Entry Local Script

```
#########################################################################
#  Job Entry Local Script
#  JobEntry.txt
#  Granite Systems
#  D. Seger  08-Dec-99
#########################################################################

# This script is designed to test the following Job Entry Commands
#       JobEntryStart
#       JobEntryComplete
#       JobEntryCancel
#       JobEntryAbort
#       JobEntryError
#       JobEntryAttributeSet
#       JobEntryAttributeGet
#       JobEntryListShow
#       JobEntryShow

##  Run this script as -->  script -scr=scripts\jobentry.txt -errorContinue=true

##  Manual set-up
        # Create a Fax FU
        # Run TOD.TXT script on the Fax FU ( script -scr=scripts\tod.txt -errorContinue=true )
        # Run this script on the Fax FU

##  Script set-up
        SfxSendSfx -todhandle=3 -life=0;            # will have jobHandle=4096 JobEntryId=1,2
        SfxSendSfx -todhandle=3 -life=0;            # will have jobHandle=4097 JobEntryId=1,2
        SfxSendSfx -todhandle=3 -life=2;            # will have jobHandle=4098 JobEntryId=1,2
        SfxSendSfx -todhandle=3 -life=0;            # will have jobHandle=4099 JobEntryId=1,2
        TodSendSfxCreate –todhandle=30 –dodhandle=2;
        TodSendSfxTsinfoSet –todhandle=30;
        SfxSendSfx -todhandle=2;                    # will have jobHandle=4100


##########  Job Status Commands  ############

JobEntryListShow
  -jobHandle=4100;

        Echo -str="   ^^^  Should have returned null list";

JobEntryShow
  -jobHandle=4096
  -jobEntryId=1;

        Echo -str="   ^^^  Should have job status code = 1";

JobEntryStart
  -jobHandle=4096
  -jobEntryId=1;

JobEntryShow
  -jobHandle=4096
  -jobEntryId=1;

        Echo -str="   ^^^  Should have job status code = 2";

JobEntryComplete
  -jobHandle=4096
  -jobentryId=1;

JobEntryListShow
  -jobHandle=4096;

        Echo -str="   ^^^  List should have 4096 - 1 2 ";

JobEntryCancel
  -jobHandle=4096
  -jobEntryId=2;

JobEntryListShow
```

```
-jobHandle=4096;

       Echo -str="   ^^^ List should have 4096 - 1 2 ";

JobCancel
  -jobHandle=4096;

JobComplete
  -jobHandle=4097;

JobListShow;

       Echo -str="   ^^^  List should have 4098 4099 4100";

JobEntryAbort
  -jobHandle=4098
  -jobEntryId=1;

JobEntryListShow
-jobHandle=4098;

       Echo -str="   ^^^ List should have 4096 - 1 2 ";

JobEntryComplete
  -jobhandle=4098
  -jobEntryId=1;

JobEntryComplete
  -jobhandle=4098
  -jobEntryId=2;

JobComplete
  -jobHandle=4098;

JobListShow;

       Echo -str="   ^^^  List should have 4098 4099 4100";

##########  Job Attribute Commands  #############

JobEntryAttributeSetInt                     # set retry count = 7
  -jobHandle=4099
  -jobEntryId=1
  -attributeId=12032
  -attributeValue=7;

JobEntryAttributeGet                # verify new attribute
  -jobHandle=4099
  -jobEntryId=1
  -attributeId=12032;

       Echo -str="   ^^^  Should show attribute 12032 = 7";

JobEntryAttributeSetEnum            # set priority = 50
  -jobHandle=4099
  -jobEntryId=1
  -attributeId=12031
  -attributeValue=50;

JobEntryAttributeGet                # verify new attribute
  -jobHandle=4099
  -jobEntryId=1
  -attributeId=12031;

       Echo -str="   ^^^  Should show attribute 12031 = 50";

JobEntryAttributeSetBool                    # set transfer timeout settable = true
  -jobHandle=4099
  -jobEntryId=1
  -attributeId=12037
  -attributeValue=true;

JobEntryAttributeGet                # verify new attribute
  -jobHandle=4099
  -jobEntryId=1
  -attributeId=12037;
```

```
        Echo -str="   ^^^  Should show attribute 12037 = True";

JobEntryAttributeSetString                  # set retry count = Granite
  -jobHandle=4099
  -jobEntryId=1
  -attributeId=12032
  -attributeValue=Granite;

JobEntryAttributeGet                # verify new attribute
  -jobHandle=4099
  -jobEntryId=1
  -attributeId=12032;

        Echo -str="   ^^^  Should show attribute 12032 = Granite";


##########  Invalid Job Commands  ############
        Echo -str="           ^^^  all commands after this point should fail  ^^^";

JobEntryStart
  -jobHandle=4099
  -jobEntryId=7;                            # invalid job entry ID

JobEntryStart
  -jobHandle=4096                           # invalid job
  -jobEntryId=1;

JobEntryComplete
  -jobHandle=4099
  -jobEntryId=7;                            # invalid job entry ID

JobEntryComplete
  -jobHandle=4096                           # invalid job
  -jobEntryId=7;

JobEntryCancel
  -jobHandle=4099
  -jobEntryId=7;                            # invalid job entry ID

JobEntryCancel
  -jobHandle=4096                           # invalid job
  -jobEntryId=7;

JobEntryAbort
  -jobHandle=4099
  -jobEntryId=7;                            # invalid job entry ID

JobEntryAbort
  -jobHandle=4096                           # invalid job
  -jobEntryId=7;

JobEntryError
  -jobHandle=4099
  -jobEntryId=7;                            # invalid job entry ID

JobEntryError
  -jobHandle=4096                           # invalid job
  -jobEntryId=7;

JobEntryAttributeSetInt
  -jobHandle=4096                           # invalid job
  -jobEntryId=1
  -attributeId=12032
  -attributeValue=1;

JobEntryAttributeSetInt
  -jobHandle=4099
  -jobEntryId=7                             # invalid job entry ID
  -attributeId=12032
  -attributeValue=1;

JobEntryAttributeGet
  -jobHandle=4099
  -jobEntryId=7                             # invalid job entry ID
  -attributeId=12032;
```

```
JobEntryAttributeGet
  -jobHandle=4096                          # invalid job
  -jobEntryId=1
  -attributeId=12032;

JobEntryAttributeGet
  -jobHandle=4096
  -jobEntryId=1
  -attributeId=12099;               # invalid attribute

JobEntryListShow
  -jobHandle=4096;                  # job was deleted

JobEntryShow
  -jobHandle=4099
  -jobEntryId=7;                           # invalid job entry ID

JobEntryShow
  -jobHandle=4096                          # invalid job
  -jobEntryId=1;

Echo -str="   ### END OF SCRIPT ###";
```

Job Entry Remote Script

```
########################################################################
#  Job Entry Remote Script
#  JobEntryRmt.txt
#  Granite Systems
#  D. Seger  08-Dec-99
########################################################################

# This script is designed to test the following Job Entry Commands
#       RmtQueryJobEntryStatus
#       RmtCancelJobEntry
#       RmtSuspendJobEntry
#       RmtResumeJobEntry
#       RmtChangeJobEntryAttribute
#       RmtStartMonitorJobStatus
#       RmtCancelMonitorJobStatus

##  Run this script as -->  script -scr=scripts\jobentryrmt.txt -errorContinue=true

##  Manual set-up
        # Create an Extended Fax FU
        # Create a Client FU
        # Open a session from the Client FU to the Extended Fax FU
        # Run TOD.TXT script on the Client FU ( script -scr=scripts\tod.txt -errorContinue=true )
        # Run this script on the Client FU


##  Script set-up
        RmtSendEfx -todhandle=4;
                # creates the following:
                #    jobhandle=4096 with jobentryhandle=1, 2
                #    jobhandle=4097 with jobentryhandle=1, 2, 3



##########  Remote Job Status Commands  ############

RmtStartMonitorJobStatus                    # subscribe to all statuses
  -jobHandle=4096
  -notificationMode={(0 1 2 3 4 5 6 7 8) true};     # include job entry statuses

RmtStartMonitorJobStatus                    # subscribe to only Cancel status
  -jobHandle=4097
  -notificationMode={(7) true};


##########  Remote Job Entry Status Commands  ############

RmtQueryJobEntryStatus
  -jobHandle=4096
  -jobEntryId=1;

        Echo -str="   ^^^  Should be job status = 1";


##########  Remote Job Entry Status Commands  ############

RmtSuspendJobEntry                          # suspend job
  -jobHandle=4096
  -jobEntryId=1;

        Echo -str="   ^^^  Should be notification of job 4096, entry 1 = 3";

RmtQueryJobStatus
  -jobHandle=4096;

        Echo -str="   ^^^  Should be job status = 3";

RmtCancelMonitorJobStatus
  -jobhandle=4096;

RmtResumeJobEntry
  -jobHandle=4096
```

```
  -jobEntryId=1;

        Echo -str="   ^^^  Should NOT be notification of job 4096, entry 2 = 7";

RmtCancelJobEntry                                   # cancel job entry - it should be deleted
since life=0
  -jobHandle=4097
  -jobEntryId=1
  -abort=true;

        Echo -str="   ^^^  Should be notification of job 4097, entry 1= 7";

RmtQueryJobEntryStatus
  -jobHandle=4096
  -jobEntryId=1;

        Echo -str="   ^^^  Should be job status = 1";

RmtCancelJob
  -jobHandle=4096
  -abort=false;


##########  Job Attribute Commands  #############

RmtChangeJobEntryAttributeInt               # set retry count = 7
  -jobHandle=4097
  -jobEntryId=2
  -attributeId=13020
  -attributeValue=7;

        Echo -str="   ^^^  on Efx FU ( JobEntryAttributeGet -jobHandle=4097 -jobEntryId=2 -
attributeId=13020 ) should return 13020 = 7";


##########  Invalid Job Commands  #############

        Echo -str="           ^^^  all commands after this point should fail  ^^^";

RmtQueryJobEntryStatus
  -jobHandle=4999                           # invalid job handle
  -jobEntryId=1;

RmtQueryJobEntryStatus
  -jobHandle=4097
  -jobEntryId=99;                                   # invalid job entry handle

RmtSuspendJobEntry
  -jobHandle=4999                                   # invalid job handle
  -jobEntryId=1;

RmtSuspendJobEntry
  -jobHandle=4097
  -jobEntryId=1;                                    # invalid job entry handle

RmtCancelJobEntry
  -jobHandle=4999                                   # invalid job handle
  -jobEntryId=1
  -abort=true;

RmtCancelJobEntry
  -jobHandle=4097
  -jobEntryId=99                                    # invalid job entry handle
  -abort=true;

RmtSuspendJobEntry
  -jobHandle=4999                                   # invalid job handle
  -jobEntryId=1;

RmtSuspendJobEntry
  -jobHandle=4096
  -jobEntryId=99;                                   # invalid job entry handle

RmtResumeJobEntry
  -jobHandle=4999                                   # invalid job handle
  -jobEntryId=1;
```

```
RmtResumeJobEntry
  -jobHandle=4096
  -jobEntryId=99;                                    # invalid job entry handle

RmtChangeJobEntryAttributeEnum
  -jobHandle=4999                                    # invalid job handle
  -jobEntryId=1
  -attributeId=13020
  -attributeValue=7;

RmtChangeJobEntryAttributeEnum
  -jobHandle=4096
  -jobEntryId=99                                     # invalid job entry handle
  -attributeId=13020
  -attributeValue=7;

RmtChangeJobEntryAttributeBool               # invalid attribute type for ID 13020
  -jobHandle=4096
  -jobEntryId=1
  -attributeId=13020
  -attributeValue=True;


Echo -str="   ### END OF SCRIPT ###";
```

DocStore Local Script

```
########################################################################
#  DocStore Local Script
#  Doc.txt
#  Granite Systems
#  D. Seger  08-Dec-99
########################################################################

# This script is designed to test the following DocStore Commands
#       DocDsOperatorIntervention
#       DocDsOperatorInformation
#       DocStoreDoc
#       DocListFolder
#       DocListFolderDoc
#       DocMoveDoc
#       DocCopyDoc
#       DocDeleteDoc
#       DocChangeDocDescription
#       DocCreateFolder
#       DocDeleteFolder
#       DocChangeFolderDescription

##  Run this script as -->  script -scr=scripts\doc.txt -errorContinue=true

##  Manual set-up
        # Create a DocStore FU
        # Create a Client FU
        # Open a Session from the Client to the DocStore FU ( sessionopen -slmip=127.0.0.1 -
fuhandle= )
        # Run the following on the Client ( RmtSubscribeEvent -dynamicStatusId=(11001 11002) -
life=1 )
        # Run this script on the DocStore FU

##  Script set-up

DodCreateFromChars
    -dodhandle=20
    -chr=30
    -chcount=1000
    -segcount=10
    -blockcount=2;

TodStoreDocCreate
    -todHandle=30
    -dodHandle=20
    -folderId=0
    -modeOfStore=2
    -ownerName=Granite
    -docComment="Document B";


##########  Doc Folder Commands  ############

DocListFolder;

        Echo -str="   ^^^   Should have returned list with no folders";

DocCreateFolder
  -ownerName="Granite Systems"
  -folderComment="Folder A";

DocCreateFolder
  -ownerName="Granite Systems"
  -folderComment="Folder B";

DocCreateFolder;

DocListFolder;

        Echo -str="   ^^^   Should have returned list with 3 folders - id=2 has no description";

DocChangeFolderDescription
  -folderId=1
```

```
  -ownerName="IBM"
  -folderComment="Folder C";

DocListFolder;

       Echo -str="   ^^^   Should have returned list with 3 folders - second has new name &
comment";

DocChangeFolderDescription
  -folderId=1
  -ownerName="New Owner Name";

DocListFolder;

       Echo -str="   ^^^   Should have returned list with 3 folders - second has new name only";

DocChangeFolderDescription
  -folderId=1
  -folderComment="New Folder Comment";

DocListFolder;

       Echo -str="   ^^^   Should have returned list with 3 folders - second has new comment
only";

DocChangeFolderDescription
  -folderId=1;

DocListFolder;

       Echo -str="   ^^^   Should have returned list with 3 folders - second has no changes";

##########  Doc Commands  ############

DocStoreDoc                          # docid=1
  #-todHandle=
  -dodHandle=20                      # earlier created DOD
  -folderId=0                # to Folder A

  #-dataSource=
  #-dataHandle=
  -modeOfStore=1
  -ownerName=Granite
  -docComment="Document A";

DocStoreDoc                          # docid=2
  -todHandle=30;

DocListFolderDoc
  -folderId=0;

       Echo -str="   ^^^   Should have returned list with two documents (ID=0,1)  Doc ID 1
should have blkcount=2";

DocStoreDoc                          # docid=3
  -todHandle=30;


DocCopyDoc
  -srcFolderId=0                     # Folder A
  -srcDocumentId=1           # Document A
  -dstFolderId=1                     # Folder B
  -updateDateTime=true;              # update time-stamp

DocMoveDoc
  -srcFolderId=0                     # Folder A
  -srcDocumentId=2           # Document B
  -dstFolderId=1                     # Folder B
  -updateDateTime=false;     # do not update time-stamp

DocMoveDoc
  -srcFolderId=0                     # Folder A
  -srcDocumentId=3
  -dstFolderId=1                     # Folder B
  -updateDateTime=true;              # update time-stamp
```

```
DocListFolderDoc
  -folderId=0;

        Echo -str="   ^^^   should return list with document ID=1";

DocListFolderDoc
  -folderId=1;

        Echo -str="   ^^^   should return list with documents ID=4,5,6";

DocChangeDocDescription
  -folderId=1                 # Folder B
  -documentId=4                       # Document A
  -ownerName="IBM"
  -docComment="Document C";

DocListFolderDoc
  -folderId=1;

        Echo -str="   ^^^   should return list with documents ID=4,5,6 and 4 has new name &
description";

DocChangeDocDescription
  -folderId=1
  -documentId=4
  -ownerName="New Name";

DocListFolderDoc
  -folderId=1;

        Echo -str="   ^^^   should return list with documents ID=4,5,6 and 4 has only new name";

DocChangeDocDescription
  -folderId=1
  -documentId=4
  -docComment="New Description";

DocListFolderDoc
  -folderId=1;

        Echo -str="   ^^^   should return list with documents ID=4,5,6 and 4 has only new
description";

DocChangeDocDescription
  -folderId=1
  -documentId=4;

DocListFolderDoc
  -folderId=1;

        Echo -str="   ^^^   should return list with documents ID=4,5,6 and 4 has no changes";


DocDeleteDoc
  -folderId=0                 # Folder A
  -documentId=1;                      # Document C

DocListFolderDoc
  -folderId=0;

Echo -str="   ^^^   should return null list";

DocDeleteFolder
  -folderId=0;

DocListFolder;

        Echo -str="   ^^^   should return list with only folder ID=1,2";

##########  Dynamic Status Commands  ############

DocDsOperatorIntervention
  -requiredAction="This is the required action string.";

#DocDsOperatorIntervention;
```

```
DocDsOperatorInformation
  -operatorInformation="This is the operator information string.";

#DocDsOperatorInformation;

        Echo -str="   ^^^   should be 2 notifications on Client FU";

##########  Failed Commands  ############
        Echo -str="   ^^^   all commands after this point should fail";

DocStoreDoc
  -todHandle=99                      # invalid TOD
  -dodHandle=20
  -folderId=0;

DocStoreDoc
  -todHandle=30
  -dodHandle=99                      # invalid DOD
  -folderId=0;

DocStoreDoc
  -todHandle=30
  -dodHandle=20
  -folderId=99;                      # invalid Folder Id

DocListFolderDoc
  -folderId=99;                      # invalid folder ID

DocMoveDoc
  -srcFolderId=99                    # invalid folder ID
  -srcDocumentId=2
  -dstFolderId=1
  -updateDateTime=false;

DocMoveDoc
  -srcFolderId=0
  -srcDocumentId=99          # invalid document ID
  -dstFolderId=1
  -updateDateTime=false;

DocMoveDoc
  -srcFolderId=0
  -srcDocumentId=2
  -dstFolderId=99                    # invalid folder ID
  -updateDateTime=false;

DocCopyDoc
  -srcFolderId=99                    # invalid folder ID
  -srcDocumentId=2
  -dstFolderId=1
  -updateDateTime=false;

DocCopyDoc
  -srcFolderId=0
  -srcDocumentId=99          # invalid document ID
  -dstFolderId=1
  -updateDateTime=false;

DocCopyDoc
  -srcFolderId=0
  -srcDocumentId=2
  -dstFolderId=99                    # invalid folder ID
  -updateDateTime=false;

DocDeleteDoc
  -folderId=99               # invalid folder ID
  -documentId=1;

DocDeleteDoc
  -folderId=0
  -documentId=99;                    # invalid document ID

DocChangeDocDescription
  -folderId=99               # invalid folder ID
  -documentId=3
  -ownerName="IBM"
```

```
  -docComment="Document C";

DocChangeDocDescription
  -folderId=1
  -documentId=99                    # invalid document ID
  -ownerName="IBM"
  -docComment="Document C";

DocDeleteFolder
  -folderId=1;               # documents still in folder

DocChangeFolderDescription
  -folderId=99               # invalid folder ID
  -ownerName="IBM"
  -folderComment="Folder C";


Echo -str="   ### END OF SCRIPT ###";
```

DocStore Remote Script

```
########################################################################
#  DocStore Remote Script
#  DocRmt.txt
#  Granite Systems
#  D. Seger  14-Nov-99
########################################################################

# This script is designed to test the following DocStore Commands
#       RmtListFolder
#       RmtListFolderDoc
#       RmtStoreDoc
#       RmtMoveDoc
#       RmtCopyDoc
#       RmtDeleteDoc
#       RmtChangeDocDescription
#       RmtCreateFolder
#       RmtDeleteFolder
#       RmtChangeFolderDescription
#       RmtRetrieveDoc
#       RmtRequestDataTransfer

##  Run this script as -->  script -scr=scripts\docrmt.txt -errorContinue=true

##  Manual set-up
        # Create a Client FU
        # Create a DocStore FU
        # Open a session from the Client FU to the DocStore FU
        #     ( sessionopen -slmip=127.0.0.1 -fuHandle= )
        # Run this script on the Client FU

##  Script set-up
        dodcreatefromchars -dodhandle=20 -chr=30 -chcount=100 -segcount=4 -blockcount=3;
        dodcreatefromfile -dodhandle=21 -filename=sfu.dll;

##########  Remote Folder Commands  ############

RmtListFolder;                # should return empty set

RmtCreateFolder
  -ownerName="Granite Systems"
  -folderComment="Folder A";

RmtCreateFolder
  -ownerName="Granite Systems"
  -folderComment="Folder B";

RmtListFolder;

        Echo -str="   ^^^ should list two folders = 0, 1";

RmtChangeFolderDescription
  -folderId=1
  -ownerName="IBM"
  -folderComment="Folder C";

RmtListFolder;

        Echo -str="   ^^^ should list two folders = 0, 1 and 1 has a new name and description";

RmtCreateFolder;                      #folder id = 2

##########  Remote DocStore Commands  ############

RmtStoreDoc                           # document ID = 1
  #-todHandle=
  -dodHandle=20                       # earlier created DOD
  -folderId=0                 # to Folder A
  #-dataSource=
  #-dataHandle=
  -modeOfStore=1
  -ownerName=Granite
  -docComment="Document A";
```

```
RmtStoreDoc                          # document ID = 2
  #-todHandle=
  -dodHandle=20
  -folderId=0                 # to Folder A
  #-dataSource=
  #-dataHandle=
  -modeOfStore=2
  -ownerName=Granite
  -docComment="Document B";

RmtListFolderDoc
  -folderId=0;

       Echo -str="   ^^^  should return list with two documents (ID=1,2)";

RmtStoreDoc                          # document ID = 3
  #-todHandle=
  -dodHandle=20
  -folderId=0                 # to Folder A
  #-dataSource=
  #-dataHandle=
  -modeOfStore=2;


RmtStoreDoc                          # document ID = 4
  #-todHandle=
  -dodHandle=20
  -folderId=0
  #-dataSource=
  #-dataHandle=
  -modeOfStore=2;

RmtStoreDoc                          # document ID = 5
  -dodHandle=21
  -folderId=0;

RmtStoreDoc                          # document ID = 6
  #-todHandle=
  -dodHandle=20
  -folderId=0
  #-dataSource={127.0.0.1 1}
  #-dataHandle=
  -modeOfStore=2;

##########  Remote Doc Handling Commands  #############

RmtCopyDoc                           # document ID = 7
  -srcFolderId=0                     # Folder A is source
  -srcDocumentId=1            # Document A
  -dstFolderId=1                     # Folder B is destination
  -updateDateTime=true;              # update time-stamp

RmtMoveDoc                           # document ID = 8
  -srcFolderId=0                     # Folder A is source
  -srcDocumentId=2            # Document B
  -dstFolderId=1                     # Folder B is destination
  -updateDateTime=false;      # do not update time-stamp

RmtMoveDoc                           # document ID = 9
  -srcFolderId=0
  -srcDocumentId=1
  -dstFolderId=2
  -updateDateTime=true;

RmtListFolderDoc
  -folderId=1;

       Echo -str="   ^^^ should have returned list with documents ID=7 8";

RmtChangeDocDescription
  -folderId=1                 # Folder B
  -documentId=7                      # Document A
  -ownerName="IBM"
  -docComment="Document C";
```

```
## List documents in folder A to verify new description

RmtListFolderDoc
  -folderId=1;

       Echo -str="   ^^^ should have returned list with documents ID=7,8 and 7 has new name &
description";

RmtDeleteDoc
  -folderId=2
  -documentId=9;

RmtListFolderDoc
  -folderId=0;

       Echo -str="   ^^^ should have returned list with document ID= 3 4 5 6";

RmtDeleteFolder
  -folderId=2;

RmtListFolder;

       Echo -str="   ^^^ should have returned list with only folder ID= 0 1";

##########  Doc Data Commands  #############


RmtRetrieveDoc                # data handle 23
  -export=true
  -folderId=1
  -documentId=7
  -startBlock=2
  -endBlock=3;

RmtRequestDataTransfer
  -dataHandle=23;


RmtRetrieveDoc                # data handle 24
  -export=false
  -folderId=1
  -documentId=8;

DodListShow;

       Echo -str="   ^^^ should list DOD's 20 21 4106 4107";

##########  Invalid Job Commands  #############
       Echo -str="           ^^^  all commands after this point should fail  ^^^";



RmtRetrieveDoc
  -export=true
  -folderId=99                # invalid folder ID
  -documentId=7;


RmtRetrieveDoc
  -export=true
  -folderId=1
  -documentId=99;                     # invalid document ID


RmtRetrieveDoc
  -export=true
  -folderId=1
  -documentId=7
  -startBlock=9                       # invalid data blocks
  -endBlock=9;

RmtChangeFolderDescription
  -folderId=99                # invalid folder ID
  -ownerName="IBM"
  -folderComment="Folder C";
```

```
RmtStoreDoc
  -todHandle=99                    # invalid TOD
  -dodHandle=20
  -folderId=0;

RmtStoreDoc
  #-todHandle=
  -dodHandle=99                    # invalid DOD
  -folderId=0;

RmtStoreDoc
  #-todHandle=
  -dodHandle=20
  -folderId=99;                    # invalid folder ID

RmtCopyDoc
  -srcFolderId=99                  # invalid folder ID
  -srcDocumentId=7
  -dstFolderId=1
  -updateDateTime=true;

RmtCopyDoc
  -srcFolderId=1
  -srcDocumentId=99          # invalid document ID
  -dstFolderId=1
  -updateDateTime=true;

RmtCopyDoc
  -srcFolderId=1
  -srcDocumentId=7
  -dstFolderId=99                  # invalid folder ID
  -updateDateTime=true;

RmtMoveDoc
  -srcFolderId=99                  # invalid folder ID
  -srcDocumentId=7
  -dstFolderId=1
  -updateDateTime=true;

RmtMoveDoc
  -srcFolderId=1
  -srcDocumentId=99          # invalid document ID
  -dstFolderId=1
  -updateDateTime=true;

RmtMoveDoc
  -srcFolderId=1
  -srcDocumentId=7
  -dstFolderId=99                  # invalid folder ID
  -updateDateTime=true;

RmtListFolderDoc
  -folderId=99;                    # invalid folder ID

RmtDeleteDoc
  -folderId=99               # invalid folder ID
  -documentId=7;

RmtDeleteDoc
  -folderId=1
  -documentId=99;                  # invalid document ID

RmtDeleteFolder
  -folderId=99;                    # invalid folder ID

RmtRequestDataTransfer
  -dataHandle=99;            # invalid data handle


Echo -str="   ### END OF SCRIPT ###";
```

Print Script

```
#########################################################################
#  Print Script
#  Print.txt
#  Granite Systems
#  D. Seger  08-Dec-99
#########################################################################

# This script is designed to test the following Print and Remote Print Commands
#       PrnDsOperationStatus
#       PrnDsErrorDetail
#       PrnDsPaperInputTrayStatus
#       PrnListPrintJob
#       PrnPrint
#       RmtListPrintJob
#       RmtPrint

##  Run this script as -->  script -scr=scripts\print.txt -errorContinue=true

##  Manual set-up
        # Create a Print FU #1
        # Create a Print FU #2
        # Create a Client FU #3
        # Open a session from #1 to #2
        # Open a session from #3 to #1
        #    sessionopen -slmip=127.0.0.1 -fuhandle=
        # Select the session on #2 as the current session
        #    SessionSelect -sessionhandle=
        # Run Tod.txt script on both #1 and #2
        #    script -scr=scripts\tod.txt -errorContinue=true
        # Subscribe the #3 FU to receive dynamic status from the #1 FU
        #    rmtsubscribeevent -dynamicstatusid=(10000) -life=1
        # On Print FU #1, set the Client FU Handle to a variable
        #    setvar -name=ClientFU  -value="<Client FU # here>"
        # Run this script on #1

##  Script set-up
        getvar -name=ClientFU;

##########  Print Status  ############

PrnDsOperationStatus
  -noPaper=true
  -noToner=False
  -doorOpen=True
  -jammed=False
  -offline=True
  -receiving=False
  -error=True
  -normal=False
  -paperNearEnd=True
  -tonerNearEnd=False
  -fatalError=True;                 # Oh boy, call a service tech!

        Echo -str="   ^^^ check for notifications on Client FU - printerStatusCode = 0 2 4 6 8 10
";

PrnDsOperationStatus
  -jammed=True;

        Echo -str="   ^^^ check for notifications on Client FU - printerStatusCode = 3";

PrnDsErrorDetail
  -statusCode=2
  -systemError="This printer is broken"
  -others="Call a technician";

        Echo -str="   ^^^ query from Client FU ( RmtQueryDynamicStatus -dynamicStatusId=10001 )";
        Echo -str="   ^^^ should state 'This printer is broken' and 'Call a technician'";

#PrnDsPaperInputTrayStatus  ### This command is not yet supported
#  -inputSelect=1                           # tray 1
#  -size=21                                 # A4
```

```
#  -direction=2                              # landscape
#  -existence=True;                  # tray is not empty


##########  Local Print Jobs  #############

PrnListPrintJob;

        Echo -str="   ^^^ should list no print jobs";

PrnPrint                                    # job 4096
  -todHandle=6                          # from Tod script
  -dodHandle=2                          # override Tod (was 1)
  -modeOfDataTransfer=1                         # delayed
#  -dataSource=
#  -dataHandle=
  -life=0
#  -notificationMode={(0 1 2 3) false)
#  -notificationScheme={127.0.0.1 1}
  -paperSize=21
  -resolution=10
  -paperDirection=1
  -copyCount=4
  -inputSelect=126
  -outputSelect=0
  -outputBinSelect=6
  -duplexModeSelect=1
  -duplexBindingMargin=10
  -faceUpModeSelect=1
  -priority=50
  -staplingSelect=0
  -fileName=printfile.txt;

PrnPrint                                    # job 4097
  -dodHandle=1;

PrnListPrintJob;

        Echo -str="   ^^^ should list print jobs 4096 4097";


##########  Remote Print Jobs  #############

RmtListPrintJob;

        Echo -str="   ^^^ should NACK - no print jobs ";

RmtPrint                                    # rmt job 4096
  -todHandle=6                          # from Tod script
  -dodHandle=2                          # override Tod (was 1)
  -modeOfDataTransfer=1                         # delayed
#  -dataSource=
#  -dataHandle=
  -life=1
#  -notificationMode={(0 1 2 3) false}
#  -notificationScheme={127.0.0.1 1}
  -paperSize=21
  -resolution=10
  -paperDirection=1
  -copyCount=4
  -inputSelect=126
  -outputSelect=0
  -outputBinSelect=6
  -duplexModeSelect=1
  -duplexBindingMargin=10
  -faceUpModeSelect=1
  -priority=50
  -staplingSelect=0
  -fileName=printfile.txt;

RmtPrint                                    # rmt job 4097
  -dodHandle=1;

RmtListPrintJob;

        Echo -str="   ^^^ should list print jobs 4096 4097";
```

```
RmtPrint                                    # rmt job 4098
  -dodHandle=1
  -life=1;

RmtPrint                                    # rmt job 4099
  -dodHandle=1
  -life=2;

RmtPrint                                    # rmt job 4100
  -todHandle=6
  -notificationMode={(0 1 2 3 4) false};

RmtSuspendJob
  -jobHandle=4100;

RmtResumeJob
  -jobHandle=4100;

       Echo -str="   ^^^ look for two notifications on other Print FU (job stat 3 & 1)";

RmtPrint                                    # rmt job 4101
  -todHandle=6
  -notificationMode={(0 1 2 3 4) false}
  -notificationScheme={127.0.0.1 $ClientFU};

       Echo -str="   ^^^ Start job on remote print ( JobStart -jobHandle=4101 ) look for
notifications on Client FU";

       Echo -str="   ^^^ Complete job on remote print ( JobComplete -jobHandle=4101 ) look for
notifications on Client FU";



##########  Invalid Job Commands  #############
       Echo -str="           ^^^  all commands after this point should fail  ^^^";


#PrnDsPaperInputTrayStatus    ### This command is not yet supported
#  -inputSelect=99                  # invalid tray
#  -size=21
#  -direction=2;

#PrnDsPaperInputTrayStatus    ### This command is not yet supported
#  -inputSelect=1
#  -size=99                                 # invalid size
#  -direction=2;

#PrnDsPaperInputTrayStatus    ### This command is not yet supported
#  -inputSelect=1
#  -size=21
#  -direction=9;                            # invalid direction

PrnPrint
  -dodHandle=99;                            # invalid DOD

RmtPrint
  -dodHandle=99;                            # invalid DOD

PrnPrint
  -todHandle=99;                            # invalid TOD

RmtPrint
  -todHandle=99;                            # invalid TOD

RmtPrint
  -todHandle=6
  -dodHandle=2
  -modeOfDataTransfer=1
#  -dataSource=
#  -dataHandle=
  -life=1
#  -notificationMode={(0 1 2 3) false}
#  -notificationScheme={127.0.0.1 1}
  -paperSize=1
  -resolution=5                             # invalid value
```

```
  -paperDirection=1
  -copyCount=100
  -inputSelect=126
  -outputSelect=99                    # invalid value
  -outputBinSelect=3
  -duplexModeSelect=3
  -duplexBindingMargin=10
  -faceUpModeSelect=2
  -priority=99                        # invalid value
  -staplingSelect=8
  -fileName=printfile.txt;

SessionClose;

       Echo -str="           ^^^ On Print FU #2, job 4098 should not exist ( PrnListPrintJob
)";

Echo -str="   ### END OF SCRIPT ###";
```

Send Fax Local Script

```
#########################################################################
#  Local Send Fax Script
#  Sfx.txt
#  Granite Systems
#  D. Seger  08-Dec-99
#########################################################################

# This script is designed to test the following Send Fax FU Commands
#       SfxDsStatus
#       SfxDsErrorStatus
#       SfxListSfxJob
#       SfxSendSfx

##  Run this script as -->  script -scr=scripts\sfx.txt -errorContinue=true

##  Manual set-up
        # Create a SendFax FU
        # Create a Client FU
        # Open a session from the Client FU to the SendFax FU
        # Subscribe the Client FU to receive dynamic status from the SendFax FU
        #   rmtsubscribeevent -dynamicstatusid=(12000) -life=1
        # Run the TOD script on the SendFax FU ( script -scr=scripts\tod.txt -errorContinue=true
)
        # Run this script on the SendFax FU


##  Script set-up
        DodCreateFromChars -dodHandle=10 -chr=30 -chCount=100 -segCount=1 -blockCount=1;
        TodSendEfxCreate -todHandle=50 -dodHandle=10;

##########  Proper usages  ############

SfxListSfxJob;

        Echo -str="   ^^^ Should have errored - no fax jobs";

SfxSendSfx
  -todHandle=50;

SfxListSfxJob;

        Echo -str="   ^^^ Should have returned list with one entry";

SfxSendSfx                                  # create a complex Sfx job
  -todHandle=3
  -modeOfDataTransfer=0
  #-dataSource=                            # not used
  #-dataHandle=                            # not used
  -life=1
  -notificationMode={(0 1 2 3 4 5 6 7 8) true}
  #-notificationScheme=
  -priority=25
  -retryCount=1;

SfxListSfxJob;                      # List SfxJobs

        Echo -str="   ^^^ Should have returned list with two entries";

SfxDsStatus                                 # Set the FU status
  -status=3;                        # 3 == suspended

        Echo -str="   ^^^  The Client FU Notification window should show FaxSendStatus=3";

SfxDsErrorStatus                            # Set the FU error status
  -systemError="You should see this error text"
  -others="You should also see this error text";

        Echo -str="   ^^^  Check error status on Client FU with --> rmtquerydynamicstatus -
dynamicstatusid=12002 ";


##########  Failed Commands  ############
```

```
        Echo -str="   ^^^   all commands after this point should fail";

SfxSendSfx
  -todHandle=99                              # invalid TOD
  -dodHandle=10;

SfxSendSfx
  -todHandle=3
  -dodHandle=99;                             # invalid DOD

Echo -str="   ### END OF SCRIPT ###";
```

Send Fax Remote Script

```
#########################################################################
#  Remote Send Fax Script
#  SfxRmt.txt
#  Granite Systems
#  D. Seger  08-Dec-99
#########################################################################

# This script is designed to test the following Send Fax FU Commands
#       RmtListSfxJob
#       RmtSendSfx

##  Run this script as -->  script -scr=scripts\sfxrmt.txt -errorContinue=true

##  Manual set-up
        # Create a SendFax FU
        # Create a Client FU
        # Open a session from the Client to the SendFax FU
        # Run TOD script on the Client ( script -scr=scripts\tod.txt)
        # Run this script on the Client FU


##  Script set-up
        DodCreateFromChars -dodHandle=10 -chr=30 -chCount=100 -segCount=1 -blockCount=1;
        TodSendSfxCreate –todHandle=50 –dodHandle=10;

##########  Proper usage's  ############

RmtListSfxJob;

        Echo -str="   ^^^ Should have failed - no fax jobs";

RmtSendSfx                                  # job handle 4096
  -todHandle=50;

RmtListSfxJob;

        Echo -str="   ^^^ Should have returned list with one entry";

RmtSendSfx                                  # job handle 4097
  -todHandle=3
  -modeOfDataTransfer=0
  #-dataSource=
  #-dataHandle=
  -life=1
  -notificationMode={(0 1 2 3 4 5 6 7 8) true}
  #-notificationScheme=
  -priority=50
  -retryCount=1;

RmtSendSfx                                  # job handle 4098
  -todHandle=3
  -life=1;                                  # job should end when session is closed

RmtSendSfx                                  # job handle 4099
  -todHandle=3
  -life=2;                                  # job should persist

RmtListSfxJob;

        Echo -str="   ^^^ Should have returned list with 4 entries";

RmtQueryJobEntryStatus
  -jobHandle=4097
  -jobEntryId=2;

        Echo -str="   ^^^ Should returned int-value = 1 ";

##########  Invalid Commands  ############
        Echo -str="           ^^^  all commands after this point should fail  ^^^";


RmtSendSfx
```

```
  -todHandle=99                              # invalid TOD
  -dodHandle=10;

RmtSendSfx
  -todHandle=3
  -dodHandle=99;                             # invalid DOD

RmtSendSfx
  -todHandle=3
  -modeOfDataTransfer=0
  -life=1
  -notificationMode={(0 1 2 3 4 5 6 7 8) true}
  -priority=99                          # invalid priority
  -retryCount=1;

RmtSendSfx
  -todHandle=99                              # invalid TOD
  -modeOfDataTransfer=0
  -life=1
  -notificationMode={(0 1 2 3 4 5 6 7 8) true}
  -priority=50
  -retryCount=1;

SessionClose;

       Echo -str="   ^^^   check Fax FU that job handle 4098 does NOT exist ( JobListShow )";


Echo -str="   ### END OF SCRIPT ###";
```

Extended Fax Local Script

```
#########################################################################
#  Extended Fax Script
#  Efx.txt
#  Granite Systems
#  D. Seger  05-Dec-99
#########################################################################

# This script is designed to test the following Efx Commands
#       EfxDsStatus
#       EfxDsErrorStatus
#       EfxListEfxJob
#       EfxSendEfx
#       EfxQueryReadInformation

##  Run this script as -->  script -scr=scripts\efx.txt -errorContinue=true

##  Manual set-up
        # Create an Extended Fax FU
        # Create a Client FU
        # Open a authenticated session from the Client FU to the Efx FU
        #    ( sessionopen -slmip=127.0.0.1 -fuHandle=1 -user={dennis pass} )
        # DO NOT --> Issue this command on the Client FU ( RmtSubscribeEfxEvent )
        # Run Tod.txt script on the Efx FU ( Script -scr=scripts\tod.txt )
        # Run this script on the Efx FU


##  Script set-up
        # none


##########  Efx Commands  ############

EfxDsStatus
  -status=4;

EfxDsStatus
  -status=3;

#       Echo -str="   ^^^   Check Client for two notifications";

EfxDsErrorStatus
  -systemError="big system error"
  -others="call Granite";

        Echo -str="   ^^^ query from Client FU ( RmtQueryDynamicStatus -dynamicStatusId=13000 )";
        Echo -str="   ^^^ should state FaxStatus=3";

        Echo -str="   ^^^ query from Client FU ( RmtQueryDynamicStatus -dynamicStatusId=13002 )";
        Echo -str="   ^^^ should state 'big system error' and 'call Granite'";

EfxListEfxJob;

        Echo -str="   ^^^ should list no Efx jobs";

EfxSendEfx
  -todHandle=4               # from Tod script
  #-dodHandle=1
  -modeOfDataTransfer=0
  #-dataSource={ 127.0.0.1 x }
  #-dataHandle=
  #-notificationMode={( 0 1 2 3) true}
  #-notificationScheme=
  -retryCount=5
  -retryInterval=6
  -queryInterval=7
  -coverSheetGeneration=true
  -pageHeaderGeneration=true
  -priority=50;

EfxListEfxJob;

        Echo -str="   ^^^ should list jobs 4096 4097";
```

```
EfxQueryReadInformation
  -todhandle=5;

EfxQueryReadInformation
  -todhandle=5
  -notificationMode={(0 1 2 3 4) true}
  -retryCount=5
  -scheduledDateTime=9811062015Z
  -scheduledAfterTime=9811062015Z
  -priority=50;

EfxListEfxJob;

        Echo -str="   ^^^ should list jobs 4096 4097 4098 4099";


##########  Invalid Job Commands  ############
        Echo -str="           ^^^  all commands after this point should fail  ^^^";

EfxSendEfx
  -todHandle=99;                    # invalid TOD handle

EfxSendEfx
  -todHandle=4
  -dodHandle=99;                    # invalid DOD handle

EfxQueryReadInformation
  -todhandle=99;                    # invalid TOD handle


Echo -str="   ### END OF SCRIPT ###";
```

Extended Fax Remote Script

```
#########################################################################
#  Remote Extended Fax Script
#  EfxRmt.txt
#  Granite Systems
#  D. Seger  14-Nov-99
#########################################################################

# This script is designed to test the following Remote Efx Commands
#       RmtRegisterUser
#       RmtUnregisterUser
#       RmtChangePassword
#       RmtListAllUserId
#       RmtListEfxJob
#       RmtSubscribeEfxEvent
#       RmtUnsubscribeEfxEvent
#       RmtRetrieveEfxData
#       RmtRetrieveEfxDocId
#       RmtPrintEfxData
#       RmtQuerySentEfx
#       RmtQueryEfxHistory
#       RmtQueryReadInformation
#       RmtInformRead
#       RmtSendEfx

##  Run this script as -->  script -scr=scripts\efxrmt.txt -errorContinue=true

##  Manual set-up
        # Create an Extended Fax FU
        # Create a 1st Client FU
        # Create a 2nd Client FU
        # Open a validated session from the Client FU to the Efx FU
        #     sessionopen -slmip=127.0.0.1 -fuHandle=XX -user={Administrator pass}
        # Run Tod.txt script on both the Client FU and the Efx FU
        #     Script -scr=scripts\tod.txt
        # Set the 2nd Client FU ID to a variable
        #     SetVar -name=FU1 -value="<enter FU handle here>"
        # Run this script on the 1st Client FU


##  Script set-up
        GetVar -name=FU1;


##########  Remote Efx User Commands  ############


RmtListAllUserId;

        Echo -str="   ^^^   should list no users ";

RmtRegisterUser
  -user={Leona pass};

RmtRegisterUser
  -user={"Fred Wilson"};

RmtRegisterUser
  -user={"Hiroshi Kato" "granite"};

RmtListAllUserId;

        Echo -str="   ^^^   should list three users - Leona, Fred, and Hiroshi";

RmtUnregisterUser
  -userId=Leona
  -forceDelete=true;

RmtListAllUserId;

        Echo -str="   ^^^   should list two users - Fred and Hiroshi";

##########  Remote Efx Subscription Commands  ############
```

```
# not yet supported

##########  Remote Efx Commands  ############


RmtListEfxJob;

       Echo -str="   ^^^ should list no Efx jobs";

RmtSendEfx                                  # jobs 4096 4097
  -todHandle=4;

RmtSendEfx                                  # jobs 4098 4099
  -todhandle=4
  -dodhandle=2
  -modeOfDataTransfer=0
  -notificationMode={(0 1 2 3 4 5 6 7 8) true}
  -notificationScheme={127.0.0.1 $FU1}
  -retryCount=5
  -retryInterval=10
  -queryInterval=60
  -coverSheetGeneration=false
  -pageHeaderGeneration=true
  -priority=50;

RmtSendEfx                                  # jobs 4100 4101
  -todHandle=4
  -notificationMode={(0 1 2 3 4 5 6 7 8) true};

RmtListEfxJob;

       Echo -str="   ^^^ should list 6 Efx jobs";

##########  Remote Efx Data Commands  ############

# not yet supported

##########  Unsupported Commands  ############

#RmtSubscribeEfxEvent;

#RmtUnsubscribeEfxEvent;

#RmtChangePassword
#  -oldPassword=pass
#  -newPassword=IBM;

#RmtRetrieveEfxData
#  -dataId=

#RmtRetrieveEfxDocId
#  -dataId=


#RmtPrintEfxData
#  -dataId=

#RmtQuerySentEfx
#  -jobHandle=
#  -jobEnrtyId=( )

#RmtQueryEfxHistory
#  -userId=

#RmtQueryReadInformation
#  -todHandle=
#  -notificationMode={127.0.0.1 fuhandle}
#  -retryCount=
#  -retryInterval=
#  -scheduledDateTime=
#  -scheduledAfterTime=
#  -priority=

#RmtInformRead
#  -dataId=
```

```
#  -hintDelete=
#  -receiverInfo=


##########  Invalid Job Commands  #############
       Echo -str="           ^^^  all commands after this point should fail  ^^^";

RmtRegisterUser
  -user={"Hiroshi Kato" "granite"};          # already registered

RmtUnregisterUser
  -userId=Kerry                              # invalid user
  -forceDelete=true;

RmtSendEfx
  -todHandle=99;                             # invalid TOD

RmtSendEfx
  -todhandle=4
  -dodhandle=2
  -modeOfDataTransfer=99             # invalid mode of transfer
  -retryCount=5
  -retryInterval=10
  -queryInterval=60
  -coverSheetGeneration=false
  -pageHeaderGeneration=true
  -priority=50;

       Echo -str="  ^^^ Run this on Efx FU -->  JobStart -jobHandle=4100 ";
       Echo -str="  ^^^  Look for one notification on 1st Client";
       Echo -str="  ^^^ Run this on Efx FU -->  JobStart -jobHandle=4098 ";
       Echo -str="  ^^^ Run this on Efx FU -->  JobEntryStart -jobHandle=4099 -jobEntryId=2 ";
       Echo -str="  ^^^  Look for two notifications on 2nd Client";

Echo -str="   ### END OF SCRIPT ###";
```

SFU-API Test Tool Cross-Reference

The Test tool's primary purpose was to test the SFU-SDK. This table correlates the test tool commands with the SFU-SDK API. The commands are grouped into several tested categories:

IBM Extension - these functions were implemented for and tested though internal IBM project development.
Object Search & Set/Get Method - these functions are access methods for internal SDK data structures. They involve a search for a designated object and an assignment or return of a particular value field. They are used extensively by the internal SDK mechanisms and also often used in the test tool mechanics.
SFU-SDK Internals - these functions are used extensively by the SDK itself , and are also exported for use by the programmer. Many of these functions are also directly used in portions of the test tool level code.
Test Tool Command - These functions have direct, or nearly direct analogs in the test tool command set. The test tool commands were primarily designed to test and exercise these functions.
Test Tool Command Response - These functions are used indirectly in formulating responses or decoding responses to particular test tool commands.
Test Tool Initialization - These functions are used in the default initialization of the test tool FU's.
Test Tool Deinitialization - These functions are used in the default deinitialization of the test tool FU's.

Function name	API ID	Test Category	Comment
sfuFuJscbGet	48	IBM Extension	These commands were implemented as extensions for the IBM project.
sfuFuJscbSet	49	IBM Extension	
sfuFuJscbClear	50	IBM Extension	
sfuFuJescbGet	51	IBM Extension	
sfuFuJescbSet	52	IBM Extension	
sfuFuJescbClear	53	IBM Extension	
sfuSndFaxEventReadInformed	165	N/A	These commands are implemented , but not testable due to the PSTN issues related to the Extended Fax FU
sfuSndFaxEventSentNotification	166	N/A	
sfuSndFaxEventReadConfirmation	167	N/A	
sfuSndFaxEventReceiptNotification	168	N/A	
sfuFuXTimeoutSet	57	Object Search & Set/Get Method	These commands are methods to set/get values in SDK nodes. They involve simply a node lookup and an assignment.
sfuJobPriorityGet	98	Object Search & Set/Get Method	
sfuJobPrioritySet	99	Object Search & Set/Get Method	
sfuJobModeOfDataTransferGet	100	Object Search & Set/Get Method	
sfuJobModeOfDataTransferSet	101	Object Search & Set/Get Method	
sfuJobLifeGet	102	Object Search & Set/Get Method	
sfuJobLifeSet	103	Object Search & Set/Get Method	
sfuDocDodHandleSet	22	Object Search & Set/Get Method	
sfuDocDodHandleGet	23	Object Search & Set/Get Method	
sfuFuNameGet	54	Object Search & Set/Get Method	
sfuFuSpoolStorageSizeGet	55	Object Search & Set/Get Method	
sfuFuUserDataGet	56	Object Search & Set/Get Method	
sfuJobDodHandleGet	96	Object Search & Set/Get Method	
sfuJobDodHandleSet	97	Object Search & Set/Get Method	
sfuTodDodHandleGet	229	Object Search & Set/Get Method	
sfuTrcMaskSet	234	Object Search & Set/Get Method	
sfuDodRefSet	12	Object Search & Set/Get Method	
sfuFuMcbGet	43	Object Search & Set/Get Method	
sfuFuNcbGet	45	Object Search & Set/Get Method	
sfuJobHandleGetByName	239	Object Search & Set/Get Method	
sfuJobNameSet	240	Object Search & Set/Get Method	
sfuJobentryHandleGetByName	241	Object Search & Set/Get Method	

Function name	API ID	Test Category	Comment
sfuJobentryNameSet	242	Object Search & Set/Get Method	
sfuJobentryDodHandleGet	243	Object Search & Set/Get Method	
sfuJobentryDodHandleSet	244	Object Search & Set/Get Method	
sfuDodHandleGetByName	245	Object Search & Set/Get Method	
sfuDodNameSet	246	Object Search & Set/Get Method	
sfuCapChk	11	SFU-SDK Internals	This function is used to type , range and bound check values received in the following commands : RmtPrivateAttributeSet , RmtJobAttributeSet , RmtJobEntryAttributeSet
sfuIoGet	69	SFU-SDK Internals	
sfuIoPut	70	SFU-SDK Internals	
sfuIoClose	71	SFU-SDK Internals	
sfuIoBreak	72	SFU-SDK Internals	
sfuSvcXevSet	204	SFU-SDK Internals	ALL RMT commands use this function (see test library stRmt.c)
sfuSvcXevWait	205	SFU-SDK Internals	ALL RMT commands use this function (see test library stRmt.c)
sfuSvcXevDeliver	206	SFU-SDK Internals	ALL RMT commands use this function (see test library stRmt.c)
sfuSvcXrcClear	209	SFU-SDK Internals	ALL RMT commands use this function
sfuSvcXTimeoutGet	211	SFU-SDK Internals	ALL RMT commands use this function to watch for transaction timeouts
sfuIoOpen	68	SFU-SDK Internals	The IO commands are integral in the operation of the following test commands : PrnPrint , SfxSendSfx , EfxSendEfx , DocStoreDoc, RmtPrint , RmtSendSfx , RmtSendEfx , RmtStoreDoc, DodCreateFromChars , DocCreateFromFile
sfuLock	3	SFU-SDK Internals	
sfuUnlock	4	SFU-SDK Internals	
sfuDodRefInc	13	SFU-SDK Internals	
sfuSvcXcbSet	202	SFU-SDK Internals	
sfuSvcXcbCall	203	SFU-SDK Internals	
sfuSvcXrcSet	207	SFU-SDK Internals	
sfuSvcXrcGet	208	SFU-SDK Internals	
sfuAttEncode	7	SFU-SDK Internals	
sfuAttDecode	8	SFU-SDK Internals	
sfuAttPrintEnc	6	SFU-SDK Internals	This function is used to decode attribute values in local and remote attribute commands of the test tool
sfuSvcXTimeoutSet	210	SFU-SDK Internals	This function is used to store the session timeout parameter specific to each functional unit. Each FU has a different Attribute ID for this parameter. It can also be set directly by a programmer - a shortcut method for setting the particular private attribute.
sfuDodRefDec	14	Test Tool Command	DodDestroy
sfuDodCreateFromChars	15	Test Tool Command	DodCreateFromChars
sfuDodCreateFromFile	16	Test Tool Command	DodCreateFromFile
sfuDodListShow	18	Test Tool Command	DodListShow
sfuDodShow	19	Test Tool Command	DodShow
sfuDocDsOperatorIntervention	20	Test Tool Command	DocDsOperatorIntervention
sfuDocDsOperatorInformation	21	Test Tool Command	DocDsOperatorInformation
sfuDocStoreDoc	24	Test Tool Command	DocStoreDoc
sfuDocDeleteDoc	25	Test Tool Command	DocDeleteDoc
sfuDocCopyDoc	26	Test Tool Command	DocCopyDoc
sfuDocMoveDoc	27	Test Tool Command	DocMoveDoc
sfuDocChangeDocDesc	28	Test Tool Command	DocChangeDocDescription
sfuDocCreateFolder	29	Test Tool Command	DocCreateFolder
sfuDocDeleteFolder	30	Test Tool Command	DocDeleteFolder
sfuDocChangeFolderDesc	31	Test Tool Command	DocChangeFolderDesc

Function name	API ID	Test Category	Comment
sfuDocListFolder	32	Test Tool Command	DocListFolder
sfuDocListFolderDoc	33	Test Tool Command	DocListFolderDoc
sfuEfxDsStatus	34	Test Tool Command	EfxDsStatus
sfuEfxDsErrorStatus	35	Test Tool Command	EfxDsErrorStatus
sfuEfxListFaxDataJob	36	Test Tool Command	EfxListEfxJob
sfuEfxSendExtFax	37	Test Tool Command	EfxSendEfx
sfuEfxQueryReadInformation	38	Test Tool Command	EfxQueryReadInformation
sfuGatGet	60	Test Tool Command	GlobalAttributeGet
sfuGatSet	61	Test Tool Command	GlobalAttributeSet
sfuGatListShow	63	Test Tool Command	GlobalAttributeListShow
sfuJatGet	73	Test Tool Command	JobAttributeGet
sfuJatSet	74	Test Tool Command	JobAttributeSet
sfuJeatGet	84	Test Tool Command	JobEntryAttributeGet
sfuJeatSet	85	Test Tool Command	JobEntryAttributeSet
sfuJobentryStart	89	Test Tool Command	JobEntryStart
sfuJobentryCancel	90	Test Tool Command	JobEntryCancel
sfuJobentryAbort	91	Test Tool Command	JobEntryAbort
sfuJobentryError	92	Test Tool Command	JobEntryError
sfuJobentryComplete	93	Test Tool Command	JobEntryComplete
sfuJobentryListShow	94	Test Tool Command	JobEntryListShow
sfuJobentryShow	95	Test Tool Command	JobEntryShow
sfuJobStart	105	Test Tool Command	JobStart
sfuJobCancel	106	Test Tool Command	JobCancel
sfuJobAbort	107	Test Tool Command	JobAbort
sfuJobError	108	Test Tool Command	JobError
sfuJobComplete	109	Test Tool Command	JobComplete
sfuJobQEnable	110	Test Tool Command	JobQEnable
sfuJobQDisable	111	Test Tool Command	JobQDisable
sfuJobQClear	112	Test Tool Command	JobQClear
sfuJobListShow	113	Test Tool Command	JobListShow
sfuJobShow	114	Test Tool Command	JobShow
sfuPatGet	121	Test Tool Command	PrivateAttributeGet
sfuPatSet	122	Test Tool Command	PrivateAttributeSet
sfuPatDel	123	Test Tool Command	PrivateAttributeSet
sfuPatListShow	125	Test Tool Command	PrivateAttributeListShow
sfuPrnDsOperationStatus	126	Test Tool Command	PrnDsOperationStatus
sfuPrnDsErrorDetail	127	Test Tool Command	PrnDsErrorDetail
sfuPrnDsPaperInputTray	128	Test Tool Command	PrnDsPaperInputTray
sfuPrnPrint	129	Test Tool Command	PrnPrint
sfuPrnListPrintJob	130	Test Tool Command	PrnListPrintJob
sfuSfxDsStatus	131	Test Tool Command	SfxDsStatus
sfuSfxDsErrorStatus	132	Test Tool Command	SfxDsErrorStatus
sfuSfxListFaxJob	133	Test Tool Command	SfxListSfxJob
sfuSfxSendFax	134	Test Tool Command	SfxSendSfx
sfuSndGetGlobalAttribute	135	Test Tool Command	RmtGetGlobalAttribute
sfuSndGetPrivateAttribute	136	Test Tool Command	RmtGetPrivateAttribute
sfuSndSetPrivateAttribute	137	Test Tool Command	RmtSetPrivateAttribute
sfuSndSuspendJobEntry	138	Test Tool Command	RmtSuspendJobEntry
sfuSndResumeJobEntry	139	Test Tool Command	RmtResumeJobEntry
sfuSndCancelJobEntry	140	Test Tool Command	RmtCancelJobEntry
sfuSndChangeJobEntryAttribute	141	Test Tool Command	RmtChangeJobEntryAttribute
sfuSndSuspendJob	142	Test Tool Command	RmtSuspendJob

Function name	API ID	Test Category	Comment
sfuSndResumeJob	143	Test Tool Command	RmtResumeJob
sfuSndCancelJob	144	Test Tool Command	RmtCancelJob
sfuSndFreeJobHandle	145	Test Tool Command	RmtFreeJobHandle
sfuSndChangeJobAttribute	146	Test Tool Command	RmtChangeJobAttribute
sfuSndQueryJobStatus	147	Test Tool Command	RmtQueryJobStatus
sfuSndNotifyJobStatus	148	Test Tool Command	RmtNotifyJobStatus
sfuSndStartMonitorJobStatus	149	Test Tool Command	RmtStartMonitorJobStatus
sfuSndCancelMonitorJobStatus	150	Test Tool Command	RmtCancelMonitorJobStatus
sfuSndQueryJobEntryStatus	151	Test Tool Command	RmtQueryJobEntryStatus
sfuSndNotifyJobEntryStatus	152	Test Tool Command	RmtNotifyJobEntryStatus
sfuSndQueryDynamicStatus	153	Test Tool Command	RmtQueryDynamicStatus
sfuSndSubscribeEvent	154	Test Tool Command	RmtSubscribeEvent
sfuSndUnsubscribeEvent	155	Test Tool Command	RmtUnsubscribeEvent
sfuSndSendFax	157	Test Tool Command	RmtSendSfx
sfuSndListFaxJob	158	Test Tool Command	RmtListSfxJob
sfuSndRegisterUserID	159	Test Tool Command	RmtRegisterUser
sfuSndUnregisterUserID	160	Test Tool Command	RmtUnregisterUser
sfuSndListAllUserID	161	Test Tool Command	RmtListAllUserID
sfuSndChangePassword	162	Test Tool Command	RmtChangePassword
sfuSndSubscribeFaxEvent	163	Test Tool Command	RmtSubscribeFaxEvent
sfuSndUnsubscribeFaxEvent	164	Test Tool Command	RmtUnsubscribeFaxEvent
sfuSndPrintFaxData	169	Test Tool Command	RmtPrintFaxData
sfuSndQuerySentFax	170	Test Tool Command	RmtQuerySentFax
sfuSndRetrieveFaxData	171	Test Tool Command	RmtRetrieveFaxData
sfuSndRetrieveFaxDocID	172	Test Tool Command	RmtRetrieveFaxDocID
sfuSndInformRead	173	Test Tool Command	RmtInformRead
sfuSndQueryFaxHistory	174	Test Tool Command	RmtQueryFaxHistory
sfuSndListFaxDataJob	175	Test Tool Command	RmtListEfxJob
sfuSndQuerySentFax	176	Test Tool Command	RmtQuerySentFax
sfuSndQueryReadInformation	177	Test Tool Command	RmtQueryReadInformation
sfuSndSendExtFax	178	Test Tool Command	RmtSendEfx
sfuSndRequestDataTransfer	183	Test Tool Command	This command is issued during the conduct of all remote data transfer operations , including operations caused by the following test commands :RmtRequestDataTransfer
sfuSndPrint	185	Test Tool Command	RmtPrint
sfuSndListPrintJob	186	Test Tool Command	RmtListPrintJob
sfuSndStoreDoc	187	Test Tool Command	RmtStoreDoc
sfuSndListFolder	188	Test Tool Command	RmtListFolder
sfuSndListFolderDoc	189	Test Tool Command	RmtListFolderDoc
sfuSndChangeFolderDesc	190	Test Tool Command	RmtChangeFolderDesc
sfuSndChangeDocDesc	191	Test Tool Command	RmtChangedDocDesc
sfuSndCreateFolder	192	Test Tool Command	RmtCreateFolder
sfuSndDeleteFolder	193	Test Tool Command	RmtDeleteFolder
sfuSndDeleteDoc	194	Test Tool Command	RmtDeleteDoc
sfuSndCopyDoc	195	Test Tool Command	RmtCopyDoc
sfuSndMoveDoc	196	Test Tool Command	RmtMoveDoc
sfuSndRetrieveDoc	197	Test Tool Command	RmtRetrieveDoc
sfuSvcOpen	198	Test Tool Command	SessionOpen
sfuSvcClose	199	Test Tool Command	SessionClose
sfuSvcListShow	200	Test Tool Command	SessionListShow
sfuSvcShow	201	Test Tool Command	SessionShow
sfuTodDestroy	212	Test Tool Command	TodDestroy
sfuTodStoreDocCreate	214	Test Tool Command	TodStoreDocCreate

Function name	API ID	Test Category	Comment
sfuTodPrintCreate	215	Test Tool Command	TodPrintCreate
sfuTodSendSfxCreate	216	Test Tool Command	TodSendSfxCreate
sfuTodSendSfxTsinfoSet	217	Test Tool Command	TodSendSfxTsinfoSet
sfuTodSendSfxCsinfoPut	218	Test Tool Command	TodSendSfxCsinfoPut
sfuTodQueryReadInformationCreate	219	Test Tool Command	TodQueryReadInformationCreate
sfuTodQueryReadInformationEfqPut	220	Test Tool Command	TodQueryReadInformationEfqPut
sfuTodQueryReadInformationEfqEntryPut	221	Test Tool Command	TodQueryReadInformationEfqEntryPut
sfuTodSendEfxCreate	222	Test Tool Command	TodSendEfxCreate
sfuTodSendEfxSuiSet	223	Test Tool Command	TodSendEfxSuiSet
sfuTodSendEfxFsiSet	224	Test Tool Command	TodSendEfxFsiSet
sfuTodSendEfxSdiPut	225	Test Tool Command	TodSendEfxSdiPut
sfuTodSendEfxSdiRcoiSet	226	Test Tool Command	TodSendEfxSdiRcoiSet
sfuTodSendEfxSdiEntryPut	227	Test Tool Command	TodSendEfxSdiEntryPut
sfuTodDodHandleSet	228	Test Tool Command	TodDodSet
sfuTodListShow	230	Test Tool Command	TodListShow
sfuTodShow	231	Test Tool Command	TodShow
sfuEfxRegisterUser	249	Test Tool Command	EfxRegisterUser
sfuEfxUnregisterUser	250	Test Tool Command	EfxUnregisterUser
sfuEfxListAllUserID	251	Test Tool Command	EfxListAllUserID
sfuFuShow	252	Test Tool Command	FuShow
sfuSndNotifyEvent	156	Test Tool Command	This command is issued to dynamic status Id subscribers when the following test commands are performed : PrnDsOperationStatus , PrnPrint , RmtPrint DocDsOperatorInformation , DocDsOperatorInterventionSfxDsStatus EfxDsStatus
sfuSndDataBlockDescription	184	Test Tool Command Response	This command is issued during the conduct of all remote data transfer operations , including operations caused by the following test commands :RmtPrint , RmtSendSfx , RmtSendEfx , RmtStoreDoc
sfuSndTransferDataBlock	181	Test Tool Command Response	This command is issued during the conduct of all remote data transfer operations , including operations caused by the following test commands : RmtPrint , RmtSendSfx , RmtSendEfx , RmtStoreDoc
sfuSndAck	179	Test Tool Command Response	RmtXXX (acks are returned for nearly all RmtXXX commands)
sfuSndNack	180	Test Tool Command Response	RmtXXX (nacks are returned for nearly all RmtXXX commands)
sfuTodChk	213	Test Tool Command Response	RmtPrint , RmtSendSfx , RmtSendEfx
sfuSndRequestNextData	182	Test Tool Command Response	This command is issued during the conduct of all remote data transfer operations , including operations caused by the following test commands :RmtPrint , RmtSendSfx , RmtSendEfx , RmtStoreDoc
sfuJatChk	75	Test Tool Command Response	RmtJobAttributeChange
sfuJeatChk	86	Test Tool Command Response	RmtJobEntryAttributeChange
sfuJobJatListChk	115	Test Tool Command Response	RmtPrint , RmtSendSfx , RmtSendEfx , RmtQueryReadInformation - The SDK performs range and bound checking on job attributes prior to queueing the jobs.
sfuPatChk	124	Test Tool Command Response	RmtPrivateAttributeSet - the SDK will perform range & bound checking on defined private attributes prior to allowing them to be set.
sfuDeinitialize	2	Test Tool Deinitialization	FU Deinitialization (see stDoc.c,stEfx.c,stClt.c ,stSfx.c,stPrn.c)
sfuFuFree	40	Test Tool Deinitialization	FU Deinitialization (see stDoc. c , stEfx.c , stClt.c , stSfx.c , stPrn.c)
sfuFuUnregister	42	Test Tool Deinitialization	FU Deinitialization (see stDoc. c , stEfx.c , stClt.c , stSfx.c , stPrn.c)
sfuInitialize	1	Test Tool Initialization	FU Initialization (see stDoc.c , stEfx.c , stClt.c , stSfx.c , stPrn.c)
sfuCapSet	10	Test Tool Initialization	FU Initialization (see stDoc.c , stEfx.c , stClt.c , stSfx.c , stPrn.c)
sfuFuAlloc	39	Test Tool Initialization	FU Initialization (see stDoc.c , stEfx.c , stClt.c , stSfx.c , stPrn.c)

Function name	API ID	Test Category	Comment
sfuFuRegister	41	Test Tool Initialization	FU Initialization (see stDoc.c , stEfx.c , stClt.c , stSfx.c , stPrn.c)
sfuGatChk	62	Test Tool Initialization	FU Initialization (see stDoc.c , stEfx.c , stClt.c , stSfx.c , stPrn.c)
sfuDatSet	65	Test Tool Initialization	FU Initialization (see stDoc.c , stEfx.c , stClt.c , stSfx.c , stPrn.c)
sfuFuMcbSet	44	Test Tool Initialization	FU Initialization (see stDoc.c , stEfx.c , stClt.c , stSfx.c , stPrn.c)
sfuFuNcbSet	46	Test Tool Initialization	FU Initialization (see stClt.c)
sfuAttPrintDec	5	Test Tool Command Response	Global / Private / Job / Jobentry AttributeSet Global / Private / Job / Jobentry AttributeGet
sfuFuTestConditionSet	58	Unsupported Command	Extension for Unsupported Test Tool Command
sfuFuSpy	59	Unsupported Command	Extension for Unsupported Test Tool Command
sfuDodWriteToFile	247	Unsupported Command	Extension For LPD Server Support
sfuTodPrintEntryPut	248	Unsupported Command	Extension For LPD Server Support
sfuCapGet	9		
sfuFuNcbClear	47		
sfuDatGet	64		This function is included for completeness (so Dat's are treated the same as Gat/Pat) , but is only used in debugging (fuShow)
sfuDatChk	66		
sfuDatListShow	67		
sfuJobGetNext	104		