FUJITSU Software BS2000 ASTI

Version V2.0F
June 2016

Readme file

1 General

The subsystem ASTI is a component that manages the new "Service" interface and performs the communications between one or more client tasks and at least one service tasks.

ASTI is part of the basic configuration and is initially only released for internal purposes. Since the commands appear on the user interface they are described in this paper.

*2 The release level is that of June 2016.

*2 Changes to release level June 2014 are marked with *1.
*1 Changes to release level July 2013 are marked with *2.

This and other current Readme files are shipped on the SoftBooks DVD and are available online at http://manuals.ts.fujitsu.com/.

2 Operating

ASTI is not a stand-alone system that directly provides a service to the customer. It is intended for subsystem developers that have to realize a type of client/server model. ASTI thereby provides the connection between the client and the server. Connections can be made in all directions (TU-TU, TPR-TU, TU-TPR and TPR-TPR). The client issues an order to the server and this is then processed by the server.
ASTI takes care of the communications between the different tasks.

ASTI takes the order and puts it into a wait queue for the service concerned. All parameters are stored. The service queries ASTI as to whether there is an order pending for it. If there is one for this service, it is forwarded to the service concerned. The service executes the function and reports the result to the client. The processes can be delivered either synchronously, i.e. the client waits for the result, or asynchronously. The client can also control how long the orders have to be held in the system before they are processed. An order is stored until it is processed. The client can also control how long the result of an order is stored.

There are three options:

1. The client does not want to receive a result; ASTI rejects the result.

2. The client wants to receive the result during the service session. The return message is held in main memory and deleted after the service session is ended.

3. The client wants to receive the result in all cases. The return message is stored on the disk. The return message is only deleted after it has been successfully delivered.

The subsystem ASTI manages the orders of the services in the files $TSOS.SYSQUE.ASTI and $*. SYS.<tsn>.<service-name>.<order-id>.
The format of these files has changed and cannot be used as of ASTI V2.0B. Therefore these files must be removed before migrate ASTI V2.0A to V2.0B. Subsystem ASTI must not be created.
$TSOS.SYSQUE.ASTI is created at start of the subsystem ASTI, if the file does not exist.

Following procedure is recommended:

```
    STOP-MAIL-SERVICE
    STOP-LWRESD                 (as of BS2000/OSD-BC V9.0)
*1  STOP-SUBSYSTEM REWAS        (as of BS2000 OSD/BC V10.0 on SE Server)
    STOP-SUBSYSTEM ASTI

    DELETE-FILE   $TSOS.SYSQUE.ASTI
    DELETE-FILE   $*. SYS.<tsn>.<service-name>.<order-id> (if existing)
```

3 Commands

3.1 Command Return Codes

All commands have SUBCODE1(), SUBCODE2() and MAINCODE() set. If ll is OK, the MAINCODE has the value CMD0001. If an error occurs the MAINCODE contains the key of the error message and the SKIP function is triggered.

The service and the client commands also set SDF-P variables. These variables are only set in positive cases.

SVTVAR-ORDERID
SVTVAR-SERVICE
SVTVAR-DATA

3.2 Service Management

These commands are responsible for controlling the service concerned. The service can be started and stopped with them. It is also possible to output the status of the service and carry out a few modifications.

3.2.1 Command: START-SERVICE

This command starts a service. A service is a batch job that is generated by ASTI and starts just one program. The program is defined by a symbol and a library. When the program terminates, the job also terminates.

```
START-SERVICE

 SERVICE-NAME = <text_4..16>

,FROM-FILE = *LIBRARY-ELEMENT(..)
                (LIBRARY = <full-filename_1..54>
                ,SYMBOL = <composed-name_1..32_with_underscore>)
                    | <c-string_1..32_with_lower-case>)
          | *BY-IMON(..)
                (LOGICAL-ID = SYSLNK | <composed-name_1..30>
                ,INSTALLATION-UNIT = <composed-name_1..30>
                ,VERSION = *STD | <version 3..7>
                ,DEFAULT-LIBRARY = <full-filename_1..54>
                ,SYMBOL = <composed-name 1..32 with underscore>)
          | *NONE(TSN = <text_4..4>)
          | *PROCEDURE(..)
                (<full-filename_1..54> | *LIBRARY-ELEMENT(..))
                    (LIBRARY = <full-filename_1..54>
                    ,ELEMENT = <composed-name 1..32_with_underscore>)

,PROCESSING-ADMISSION = *FROM-ORDER
                | *FROM-START
                | *EXPLICIT(..)
                     (USERID = <name_1..8>
                    ,ACCOUNT = <text_1..8>
                    ,PASSWORD = *NONE | <secure1..8>)
```

```
                ,ALLOWED-USER = *ALL
                              | *NONE
                              | *START-USER-IDENTIFICATION(..)
*2                                  (ALTERNATE-ADMIN = *NONE | list_poss(8): <name_1..8>)
                              | *BY-GUARD(NAME = <full-filename_1..24>)

                ,ORDER-RECOVERY = *PARAMETER(..)
                                  (ALLOWED = *NO | *SESSION-WIDE | *PERMANENT
                                  ,DEFAULT = *NO | *SESSION-WIDE | *PERMANENT)

                ,ORDER-LIMIT = 255 | <integer_1..32000>

                ,NUMBER-OF-TASKS = 1 | <integer_1..16>
                                  (MINIMAL = 0 | <integer_0..16>
                                  ,MAXIMAL = 16 | <integer_1..16>)

                ,START-PARAMETER = *NONE | <c-string_1..1800> | <x-string_1..3600>

                ,NOT-ALLOWED-CMD = *NONE
                                 | list-poss(5): *STOP-SERVICE | *MODIFY-SERVICE-PARAM |
                                                 *SEND-ORDER | *DELETE-ORDER |
                                                 *REQUEST-ORDER-RESULT

                ,NOT-ALLOWED-SVC = *NONE |
                                 | list-poss(5): *SVTSTOP | *SVTMODI |
                                                 *SVTSORD | *SVTDORD | *SVTRORD

                ,SYSTEM-OUTPUT = *ALL | *NONE | *STDOUT | *PARAMETER(..)
                               (SYSLST-OUTPUT = *STDOUT | *NONE | *PRINTER | *MAIL
                               ,SYSOUT-OUTPUT = *STDOUT | *NONE | *PRINTER | *MAIL )

*2              ,MONJV = *NONE | <full-filename_1..24>
```

SERVICE-NAME	a name that is unique for this system. Services beginning with $ can only started by TSOS
FROM-FILE	this parameter supplies the load library and the symbol to be loaded. Either direct input *LIBRARY-ELEMENTor indirect input via the installation monitor *BY-IMON are possible. The option *NONE can be used to make a running dialog task be seen as a service task.
*LIBRARY-ELEMENT	
LIBRARY	name of a module library.
SYMBOL	name of a CSECT or an ENTRY with which a module is loaded and started.
*BY-IMON	
LOGICAL-ID	logical name of the module library.
INSTALL-UNIT	name of a product
VERSION	version of a product. *STD is the highest installed version.
DEFAULT-LIB	default library name, if the product is not installed.
SYMBOL	name of a CSECT or an ENTRY with which a module is loaded and started.
*NONE	
TSN	a TSN of a dialog task can be input here

	*PROCEDURE	
	FILENAME	Name of a procedure file
	LIBRARY	Name of a module library.
	ELEMENT	Name of J-element in a library
	PROCESSING-ADMISSION	regulates which user environment the service works in.
	*FROM-ORDER	the service takes over the user environment of the task that sent the order. This function is not allowed for FROM-FILE=*NONE.
	*FROM-START	the service contains a user ID that is not changed during operation.
	*EXPLICIT	Start User-ID of the service
	USERID	<text_1..8> the specified user ID is taken.
	ACCOUNT	<text_1..8> account number of the user ID.
	PASSWORD	*NONE no password assigned. <secure1..8> password of the user ID.
	ALLOWED-USER	controls access to the service.
	*START-USER-IDENTIFICATION()	only the same user ID as with START-SERVICE is allowed, or one of the alternate user ID's from the list.
*2	ALTERNATE-ADMIN	a list of up to 8 (different) user ID's which also can manage the service
	*BY-GUARD	the user IDs are permitted via a guard. The guard is in the start user ID. The length of the guard is (1..8 Char).
	*NONE	the service uses only the service management.
	*ALL	all tasks have access.
	ORDER-RECOVERY	the value range of the ORDER-RECOVERY parameter in the SEND-ORDER command can be influenced with this parameter.
	ALLOWED	the value range of the ORDER-RECOVERY parameter in the SEND-ORDER command can be limited with this parameter.
	*NO	the ORDER and the RESULT are only held in main memory. If a task or the service dies, the orders concerned are acknowledged negatively. If the order task is missing, the results are deleted.
	*SESSION-WIDE	the ORDER is held in memory until the order has been processed. Unprocessed orders and results are held in main memory and only deleted after SESSION-END (/SHUTDOWN or /STOP-SUB ASTI).
	*PERMANENT	the ORDER and the RESULT are stored on the disk. Each part is only deleted after the respective action is completed. With this function, incorrect programming can create order cadavers that remain in the order user ID in the form of files.
	DEFAULT	the ORDER-RECOVERY default value is set with this parameter.

ORDER-LIMIT	this defines the maximum number of ORDERS being processed in one time.
NUMBER-OF-TASKS	1-16 number of service tasks that work for this service.
*MINIMAL	0-16 the minimal number of service tasks, which would be changed by /MODIFY-SERVICE-PARAMETER.
*MAXIMAL	1-16 the maximal number of service tasks, which would be changed by /MODIFY-SERVICE-PARAMETER.
START-PARAM	this parameter must be queried by the service after the service task is started. The record is used for initializing the service.
NOT-ALLOWED-CMD	specific commands can be blocked for a service with this parameter.
*STOP-SERVICE	This service cannot be stopped with this command.
*MODIFY-SERVICE-PARAM	this service cannot be modified.
*SEND-ORDER	no order can be placed to this service.
*DELETE-ORDER	no orders can be deleted.
*REQUEST-ORDER-RESULT	no results can be fetched.
NOT-ALLOWED-SVC	program calls for a service can be blocked with this parameter.
*SVTSTOP	this service cannot be stopped by the program.
*SVTMODI	this service cannot be modified by the program.
*SVTSORD	no orders can be placed by the program.
*SVTDORD	no orders can be deleted by the program.
*SVTRORD	no results can be fetched by the program.

Note: The TPR interface is not subject to any of these restrictions

SYSTEM-OUTPUT setting parameter of EXIT-JOB

*ALL SYSLST and SYSOUT are printed
*NONE no output
*STDOUT Behavior specified by CLASS-II-Option “SSMOUT”

*PARAMETER
SYSLST-OUTPUT Parameter for SYSLST
SYSOUT-OUTPUT Parameter for SYSOUT
*STDOUT Class II Option “SSMOUT”
*NONE no output of selected file
*PRINTER print of selected file
*MAIL sending file as mail

*2 MONJV The name of a JV to monitor the task.
*2
*2 *NONE MONJV is not used here

Note: there can be only one task being monitored by a MONJV.

3.2.2 Command: STOP-SERVICE

This command stops a service. The service is set to termination mode. The separate tasks must stop themselves. New orders are no longer accepted.

Existing order are handled according to the save order value

STOP-SERVICE SERVICE-NAME= <text_4..16> Name of service

3.2.3 Command: SHOW-SERVICE-STATUS

The status of a single service or of all services are displayed with this command. This information can also be shown into OPS-variables.

SHOW-SERVICE-STATUS SERVICE-NAME = *ALL | <text_4..16>

Output SERVICE-NAME=*ALL

```
SID    SERVICE-NAME    USERID    STATUS     TASK ALLOWED-USER PRO-ADM
@006 SERVICE3          ASTITEST RUNNING    0001 ALL               FROM-START
@005 SERVICE2          TSOS       RUNNING    0001 START-USERID  FROM-ORDER
@004 SERVICE1          TSOS       RUNNING    0001 ALL               FROM-ORDER
@003 MAILCLNT          TSOS       RUNNING    0001 ALL               FROM-START
@002 LWRESD            TSOS       RUNNING    0001 START-USERID  FROM-START
@001 $ASTIQUE          TSOS       RUNNING    0001 ALL               FROM-START
```

Output SERVICE-NAME=MAILCLNT

```
SID   SERVICE-NAME    STATUS       ADMINSTR ALLOWED-USER PROCESSING MODE
@003 MAILCLNT          RUNNING      TSOS        ALL               FROM-START SERVICE
ORDER-RECOVERY:      ALLOWED = PERMANENT    DEFAULT = PERMANENT
SERVICE-TASKS :      01                LIMITS (MIN=00,MAX=16)
SYSTEM-OUTPUT :      SYSOUT  = *NONE       SYSLST  = *NONE
SYMBOL       :       MSSRVC
LIBRARY      :       :SQN6:$TSOS.SYSLNK.MAIL.032.BACKEND
GUARD        :       *NONE
TASK: 9UKX ACT-ORDER: 000 TERMINATION: NONE       START-PARAM: NONE
%  SVTS000 Service MAILCLNT: Command executed
```

3.2.4 Command: MODIFY-SERVICE-PARAMETER

```
MODIFY-SERVICE-PARAMETER

 SERVICE-NAME = <text_4..16>

,ALLOWED-USER = *UNCHANGED
                | *ALL
                | *START-USER-IDENTIFICATION
                | *BY-GUARD(NAME=<full-filename_1..24>)

,NUMBER-OF-TASKS = *UNCHANGED | *PAST-VALUE | <integer_0..16>

,ORDER-LIMIT = *UNCHANGED | <integer_1..32000>

,START-PARAMETER = *UNCHANGED | *NONE
                   | <c-string_1..1800> | <x-string_1..3600>
```

SERVICE-NAME	Name of service
ALLOWED-USER	controls who can use the service
NUMBER-OF-TASKS	
0-16	number of service tasks that can work with the same SERVICE name. The number of tasks can be reduced to 0. In this case, the service is still present and accepts orders, but does not execute them. This parameter is ignored with a dialog service.
*PAST-VALUE	sets the number to the value before the last /MODIFY-SERVICE-PARAMETER
ORDER-LIMIT	the number of orders for this service that are in the system can be limited with this parameter.
START-PARAMETER	the initialization parameters can be changed with this parameter. The next PROCESS-ORDER receives the RC: “new parameter”. If a PROCESS-ORDER is in the wait queue when this parameter is modified, RC: 00040010 is only set after the queue timeout.

3.3 Client Commands

These commands connect the client to the service.

3.3.1 Command: SHOW-ORDER-STATUS

Information about orders is output with this command. The command works asynchronously to normal operation and consistency between the number of orders and the separate sub-queues is therefore not assured.
This information can also be shown into OPS-variables

SHOW-ORDER-STATUS

```
 INFORMATION =  *SUMMARY ( SERVICE-NAME = *ALL | <text_4..16> )
                *ORDER-LIST ( SERVICE-NAME = *ALL | <text_4..16> )
                *ALL ( ORDER-ID = < x-text_1..16 > )
```

INFORMATION:

*SUMMARY	number of orders per subqeue
*ORDER-LIST	all relevant order IDs, including status
*ALL	Informationen about the orders (administrator about all orders, normal user about his own)
SERVICE-NAME	name of the service
ORDER-ID	order identification during the same subsystem session the last 8 bytes are sufficient, for orders of other sessions all 16 bytes are required.

Output information *SUMMARY

SERVICE	ALL-Q	RDY-Q	ACT-Q	RES-Q	WAI-Q	NRR-Q	IAC-Q	DEQ-R
SERVICE4	3	1	1	1	0	0	0	0

ALL-Q	all orders of this service
RDY-Q	all orders that are waiting for execution by the service
ACT-Q	all orders that are currently being executed
RES-Q	all results that have not been fetched yet
WAI-Q	all orders that have been deferred
NRR-Q	all results that have not been requested
AC-Q	orders from stopped service (session-wide)
DEQ-R	results whose tasks no longer exist

Output information *ORDER-LIST

ORDER-ID	SERVICE	QUEUE	TASK
1266CAAA00000007	SERVICE4	READY-QUEUE	0BA4
1266CAAA00000006	SERVICE4	ACTIVE-QUEUE	0BA4
1266CAAA00000005	SERVICE4	RESULT-QUEUE	0BA4

```
%  SVTS000 Service SERVICE4: Command executed
```

Output information *ALL

```
ORDER                SERVICE      CURRENT QUEUE
1266CAAA00000007 SERVICE4     READY-QUEUE
USERID  ACCOUNT   TSN          USER-STATE
TSOS    ADMINSTR  0BA4         RESULT-REQUESTED
```

Comment:

USERID	User Id of the sender
ACCOUNT	account of the senders
TSN	TSN of the sendertask
USER-STATE	state of result request

3.3.2 Command: DELETE-ORDER

Either one order or all orders of a service can be deleted with this command. With a normal user, only orders with his user ID are deleted and with the administrator, all orders of the service concerned.

DELETE-ORDER

```
ORDER-ID = *TASK(SERVICE-NAME = <text_4..16>)
         |*USER(SERVICE-NAME = <text_4..16>)
         |*ALL(SERVICE-NAME = <text_4..16>)
         |<x-text_1..16>
```

ORDER-ID	order identification during the same subsystem session the last 8 bytes are sufficient, for orders of other sessions all 16 bytes are required. Only orders from the same task are deleted Exception: permanent orders, whose callers no longer exist.

*TASK(SERVICE-NAME=)	all orders that this task sent to the service are deleted, including permanent orders.
*USER(SERVICE-NAME=)	all orders that were sent to the service by this user ID are deleted, including permanent orders.
*ALL(SERVICE-NAME=)	all orders that were sent to the service by this user ID are deleted; with the administrator, all orders are deleted.

Note:
All orders concerned are searched for and marked with "order to delete". Orders that are currently being processed by the service are completed by the service. If the client is waiting, the result is delivered. If the client is not waiting, the result is deleted. Orders in the ready queue or in the wait queue are deleted. If the client is waiting for the result, ASTI delivers a negative result (key: SVTS903).

3.3.3 Command: REQUEST-ORDER-RESULT

The result of an asynchronous order is fetched with this command. The result is thereby deleted in the ASTI management. Results of permanent orders whose caller task no longer exists can also be fetched with this command.

REQUEST-ORDER-RESULT

```
ORDER-ID= <x-text_1..16>
        | *TASK(SERVICE-NAME=<text_1..16>)
        | *USER(SERVICE-NAME=<text_1..16>)
        ,WAIT-FOR-RESULT = *YES (TIME-LIMIT= *NO
                                | <integer_15..65535>)
                                | *NO
```

ORDER-ID	order identification during the same subsystem session the last 8 bytes are sufficient, for orders of other sessions all 16 bytes are required. Only orders from the same task are deleted Exception: permanent orders, whose callers no longer exist.
*TASK	the next order of this task is requested
*USER	the next order of this user ID is requested
SERVICE-NAME	selects the service
WAIT-FOR-RESULT	specifies, if the client waits for the result. The TIME-LIMIT specifies the time for waiting

With the *USER parameter, results whose task no longer exists can be fetched

3.3.4 Command: SEND-ORDER

An order is placed with this command.

SEND-ORDER

```
 SERVICE-NAME = <text_4..16>
                      ,WAIT-FOR-RESULT = *YES (TIME-LIMIT = *NO
                                              | <integer_15..65535>)
                                              | *NO (RESULT = *NO | *YES )
                      ,ORDER-RECOVERY = *STD
                                              | *NONE
                                              | *SESSION-WIDE
                                              | *PERMANENT
                      ,DATA = *NO | <c-string 1...1800> | <x-string 1..3600>
```

SERVICE-NAME	name of the service
WAIT-FOR-RESULT	
*YES	the order is accepted and the task waits until the order is completed. The result is displayed
TIME-LIMIT	TIME-LIMIT restricts the waiting time.
*NO	the order is accepted and processed asynchronously.
RESULT=*NO	no result requested
RESULT=*YES	a result is requested, that must be fetched later with /REQUEST-ORDER-RESULT.
ORDER-RECOVERY	
*STD	the default value of /START-SERVICE is taken.
*NONE	no special measures are met regarding transaction security.
*SESSION-WIDE	the order is retained until acknowledged by the service and the result is also retained until delivery to the client. If these events do not occur, the order or the result is deleted when the session ends.
*PERMANENT	in contrast to *SESSION-WIDE, neither orders nor result are deleted at termination of the service or the subsystem ASTI. Results can be fetched at next start of ASTI session, orders are delivered at start of the service concerned.
DATA	the contents of the parameter represent the order to the service task.

3.4 Service Command

3.4.1 Command: PROCESS-ORDER

The service fetches orders from ASTI and returns results with this command.

PROCESS-ORDER

```
ACTION=   *GET-ORDER (WAIT-FOR-ORDER =
                      *YES(TIME-LIMIT = *NO | <integer_15..65535>)
                      | *NO

          *SEND-ACK (ORDER-ID=<x-text_1..16>
                    ,RETURN-DATA = *NONE
                                   | <c-string_1..1800>
                                   | <x-string_1..3600>)

          *SEND-NAK (ORDER-ID = <x-text_1..16>
                    ,RETURN-KEY = <text_7..7>)
                    ,RETURN-DATA = *NONE
                                   | <c-string_1..1800>
                                   | <x-string_1..3600>)

          *SEND-RETRY(ORDER-ID = <x-text_1..16>
                      ,WAIT-TIME = <integer_1..2000>
                      ,RETURN-DATA = *NONE
                                     | <c-string_1..1800>
                                     | <x-string_1..3600>)

          *GET-START-PARAMETER
```

*GET-ORDER	request a new order.
WAIT-FOR-ORDER	
*YES	the service task waits until a new order arrives.
*NO	the service task checks whether a new order is pending and continues working immediately.
*SEND-ACK	order with ORDER ID was executed correctly.
RETURN-DATA	return parameter
*NONE	no value is returned
*SEND-NAK	order with ORDER ID was aborted with an error.
RETURN-KEY	specifies the error
RETURN-DATA	return parameter
*NONE	no value is returned
*SEND-RETRY	the order can temporarily not be executed.
WAIT-TIME	WAIT-TIME = time in seconds. Order should be restarted in n seconds. The wait time may extend up to the queue timeout (30 seconds).
RETURN-DATA	return parameter
*NONE	no value is returned
*GET-START-PARAMETER	initialization parameters are queried after the service start or after a request via RC.

.

4 Macros

There are also macros available for the interfaces.

5 Procedures

5.1 Reset of a central lock in ASTI

A deadlock situation occurs, if a task, which holds a lock on central data of ASTI, terminates abnormally, so that the lock is not released. As long as this lock is not released, ASTI doesn't accept new orders or perform existing orders.

As long as there is no collaboration between task termination processing and ASTI, this problem cannot be avoided. To remove the deadlock situation and to avoid the reboot of the system, an "unlock" procedure **SYSPRC.ASTI.020.RESET-LOCK** is delivered as of ASTI V2.0C.

The procedure requires the TSOS privilege.
It should be called, when a deadlock situation is suspected.
The procedure displays a corresponding message, but does not perform any modifications.

SYSPRC.ASTI.020.RESET-LOCK performs the following operations:
- Get the content of the lock area in the Asti central lock table
- Depending on the lock content one of the following messages is displayed

  1. No deadlock situation: X'00000000' or X'FFFFFFFF', nothing must be done
     returned message: "ACTIVE-Q NOT LOCKED AT THE MOMENT"

  2. With any other content the following messages are displayed:
     "ACTIVE-Q LOCKED BY TSN <TSN>"
     RESET LOCK FOR EXISTING TSN <tsn>? (Y/N) or
     RESET LOCK FOR UNKNOWN TSN <tsn>? (Y/N)

- If "Y" given, the program HELGA is called to reset the lock.

5.2 Example of a Service procedure

```
/SET-PROCEDURE-OPTION
/BEGIN-PARAMETER-DECLARATION
/END-PARAMETER-DECLARATION
/&*
/&* DO.SERVICE
/&* THIS PROCEDURE REALIZES A SERVICE
/&*
/ PROCESS-ORDER ACTION=*GET-ORDER(WAIT-FOR-ORDER=*YES)
/ IF-CMD-ERROR;END-IF
/ IF (MAINCODE() NE 'SVTS016')
/   IF-BLOCK-ERROR;END-IF
/   IMPORT-VARIABLE VARIABLE-NAME=SVTVAR-ORDERID
/   IMPORT-VARIABLE VARIABLE-NAME=SVTVAR-SERVICE
/   IMPORT-VARIABLE VARIABLE-NAME=SVTVAR-DATA
/LOOP: REPEAT
/     SHOW-VARIABLE VARIABLE-NAME=SVTVAR-SERVICE
/     IF (MAINCODE() EQ 'CMD0001')
/       PROCESS-ORDER ACTION=*SEND-ACK,ORDER-ID=&(SVTVAR-
ORDERID), -
/         RET-DATA = 'OKAY'
/       IF-BLOCK-ERROR;END-IF
/     ELSE-IF (MAINCODE() EQ 'SVTS011')
/       PROCESS-ORDER ACTION=*GET-START-PARAMETER
/       IF-BLOCK-ERROR;END-IF
/       WRITE-TEXT TEXT='START-PAR: SVTVAR-DATA'
/     ELSE-IF (MAINCODE() NE 'SVTS016')
/       WRITE-TEXT TEXT='ERROR MAINCODE = &(MAINCODE())'
/       EXIT-BLOCK BLOCK=LOOP
/     END-IF
/     PROCESS-ORDER ACTION=*GET-ORDER(WAIT-FOR-ORDER=*YES)
/     IF-CMD-ERROR;END-IF
/   UNTIL (MAINCODE() EQ 'SVTS016')
/ END-IF
[/EXIT-PROC]
/&* must be replaced to avoid CMD2001 during task termination
/EXIT-JOB
```