1 Introduction

SNS (SPOOL Notification Service) is a BS2000 subsystem that provides a tool for sending and managing notifications in the frame of the BS2000. SNS has been introduced in SPOOL, RSO, and DPRINT for the notification on print jobs, but it can be used for any BS2000 product as well as for any BS2000 application.

In SNS V1.0B the notification by mail (MAILTO and OPGMAIL),
by procedure (PROCEDURE) or in a file (FILE) is provided.

1.1 Target group

This manual is intended for nonprivileged users and BS2000 system support personnel.

1.2 Summary of contents

The manual describes the SNS subsystem and is subdivided into the following chapters:

“Introduction”
Contains a short description of SNS and provides notes on the use of the manual.

“Overview and principle of operation”
Provides an overview of SNS. An introduction to SNS is given, the notification resources and the components of SNS are explained.

“Installation and configuration”
Describes the software requirements needed for SNS and what must be considered during the installation and configuration of the subsystem.

“Definition of notification resources - example”
Contains an example of the general notification mechanism as it works for Spool & Print. It describes what the product that supports SNS must provide and what system administrator and nonprivileged user must do.

“Implementation of the notification support in a BS2000 product or application”
Explains how to implement the support of the Notification Service in a BS2000 product or application.

“Notification delivery methods”
Describes the notification delivery methods that are provided with SNS.

“SNS commands and statements”
Provides an overview of the SNS commands and the Notification Resources Manager statements.

“Notification Service APIs”
Describes the APIs that are provided with SNS

“SNS messages”
Describes the messages that are provided with SNS

At the end of the manual, you will find a glossary, list of related publications and index.

README file

Information on functional changes and additions to the current product version described in this manual can be found in the product-specific README file.
You will find it under the file name `SYSRME.SNS.010.E`.
The user ID under which the README file is cataloged can be obtained from systems support. With IMON you can also find out the file name using the following command:

```
/SHOW-INSTALLATION-PATH INSTALLATION-UNIT=SNS,LOGICAL-ID=SYSRME.E
```

You can view the README file using the `/SHOW-FILE` command or an editor, and print it out on a standard printer using the following command:

```
/PRINT-DOCUMENT $userid.SYSRME.SNS.010.E,LINE-SPACING=*BY-EBCDIC-CONTR
```

1.3 Changes since the last version of the manual

This manual for SNS V1.0B contains the following important changes which have been implemented since the previous version of the manual:

- The chapter “Installation and configuration” has been brought up to date.
- The README file of SNS V1.0A has been incorporated (notification method OPGMAIL).
- The description of the new notification methods PROCEDURE and FILE have been added.
- The new CHANGE-FILE-NOTIFICATION command has been added.

Incompatibilities

As of SNS V1.0B all the data related to the notification methods are located in the library `$SYSSNS.SYSLIB.SNRTP.METHOD` after installation.
In SNS V1.0A the libraries `SYSLIB.SNRTP.010.METHOD` and `SYSLIB.SNRTP.010.OPGMAIL` were used.

If SNS V1.0A is installed, the customer should bring his configuration files (and own methods, if any) from the libraries `SYSLIB.SNRTP.010.METHOD` and `SYSLIB.SNRTP.010.OPGMAIL` to the new library `$SYSSNS.SYSLIB.SNRTP.METHOD` using LMS.

1.4 Notational conventions

The following notational conventions are used:

Important note concerning the text. Be careful when you see this symbol.

Note

The word “Note” before an indented paragraph indicates that the paragraph contains important information.

“Reference”

References to chapters, sections or other manuals are enclosed in quotation marks.

Bold

In explanations of syntax presentations, the lines that are currently being discussed are printed in semi-bold.
Moreover, in the case of syntax presentations, the rules described in the corresponding chapters of the reference section apply.

```
SYNTAX/Example
```

Syntax presentations and example inputs and outputs are highlighted by means of a different font. Syntax presentations are also surrounded by a frame.

```
[ ]
```

Square brackets in syntax presentations: the characters within the brackets may be omitted.

1.5 Brief description of the Spool & Print Services

The Spool & Print Services for BS2000/OSD consist of various subsystems and utilities. The figure below presents the components of the Spool & Print Services.

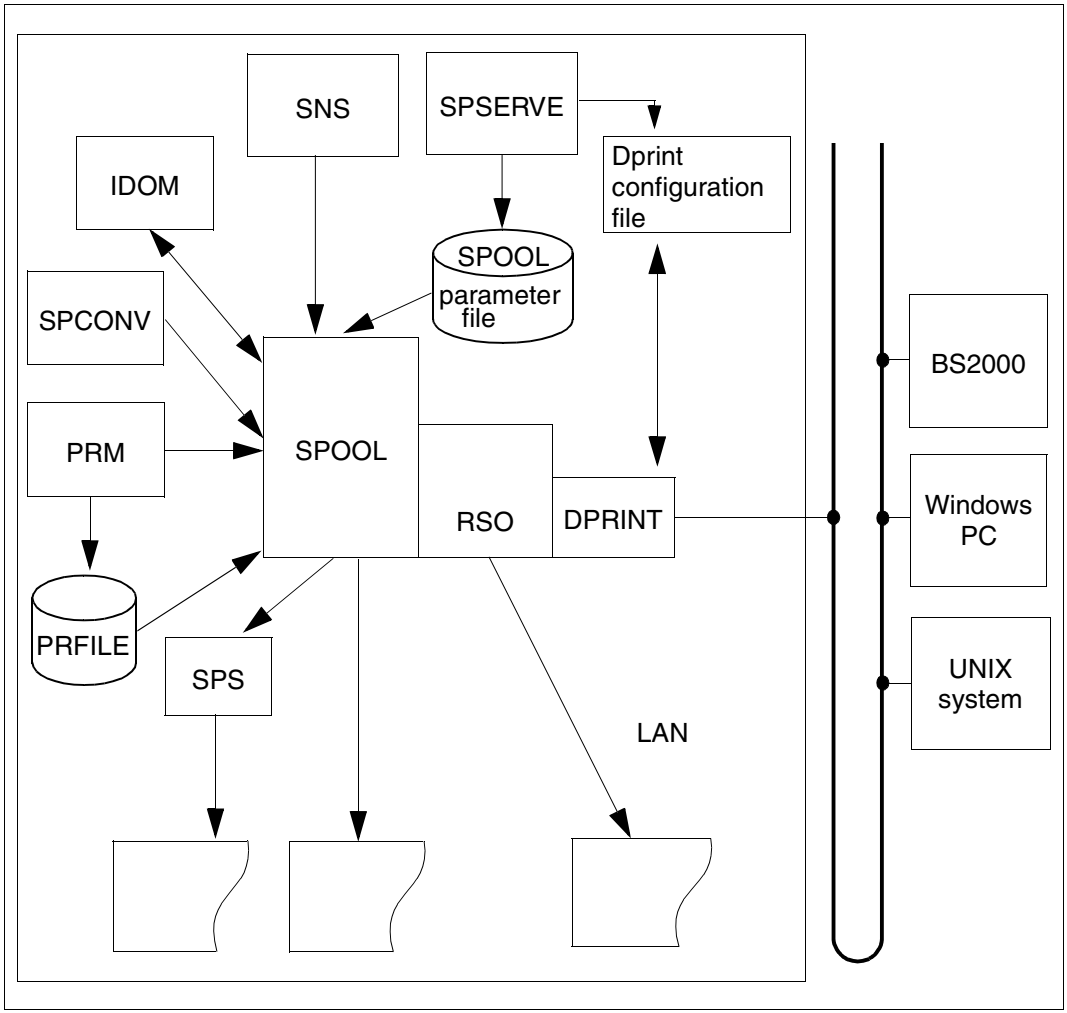

Figure 1: Overview of the Spool & Print Services

Spool & Print subsystems

SPOOL

The local SPOOL (**S**imultaneous **P**eripheral **O**peration **O**n-**L**ine) is a BS2000 subsystem. It controls the asynchronous output to printers and magnetic tapes. SPOOL is the underlying component for the other products described below which cannot run without it.

RSO

Remote **S**pool **O**utput controls output to remote printers (RSO printers) which are connected to remote computers or to the LAN. RSO only supports point-to-point data transfer.

Dprint

Distributed **Print** Services also controls output to printers that are connected to remote computers. However, to do this it uses the target computer's local SPOOL and therefore assumes control of the print output.

SPS

SPOOLAPA **P**rinting **S**ystem can be connected to the SPOOL system as a subsystem and performs printer driver tasks for APA printers.

IDOM

Integrated **D**ocument and **O**utput **M**anagement is a subsystem of the Spool & Print Services on BS2000/OSD. It extends the Spool & Print Services by providing document management functionality.

Additional administration utilities

PRM

The **P**rint **R**esource **M**anagement is used to create and manage SPOOL print resources.

SNS

The **S**POOL **N**otification **S**ervice provides a tool for sending and managing notifications in the frame of BS2000.

SPSERVE

Systems support personnel can use SPSERVE to enter RSO device managers and non-privileged users in a SPOOL parameter file where they can then edit them, delete them or output them. However, the scope of this functionality is restricted.

SPCONV

The SPOOL-CONVERTER implements the filter mechanism for printing in distributed, heterogeneous, environments with BS2000, UNIX- and PC systems, i.e. it uses filters to convert documents and print resources into a format that can be printed at the printer in question.

2 Overview and principle of operation

This chapter provides an overview of the SNS subsystem. An introduction to the Notification Service is given, the notification resources and the components of SNS are explained. Also, topics such as privileges, messages, and tasking are discussed.

2.1 Introduction to SNS

SNS (SPOOL Notification Service) is a tool for sending and managing notifications in the frame of BS2000. A first application of this notification service has been introduced in SPOOL, RSO, and DPRINT for the print job notifications, but this subsystem can be used for any BS2000 product as well as for any BS2000 application.

When a product or application wishes to offer the notification support to its users, the following steps are necessary:

- Identify the notification resources
  - identify the objects of this product or application for which a notification should be provided (e.g. PRINTJOB or PRINTER in the frame of Spool & Print)
  - identify the events relative to these objects that a user can be notified of (e.g. PRINTJOBACCEPTED, PRINTJOBCOMPLETED in the frame of Spool & Print)
  - identify the notification delivery method that will be used for the notification (e.g. the MAILTO delivery method)

  See section “Notification resources” on page 10 for a description of the notification resources.
- Include the notification resources in the notification resources file

  With the program Notification Resources Manager the notification resources must be included in the SNS data repository (notification resources file). Only privileged users (users with privilege TSOS or NOTIFICATION-MANAGER) are allowed to include definitions of resources in the notification resources file.
  See chapter “Definition of notification resources - example” on page 27 for an example.

- Prepare the product or application for the support of notifications
  - anchor in the code implementation of the product or application the raise event processing that informs SNS of the occurrence of the event. A specific macro SNPREV is provided for this purpose. Specific object attributes can be associated to the event.
  - document the events and attributes available to the end users in the product or application manuals.

  See chapter “Implementation of the notification support in a BS2000 product or application” on page 37 for a more detailed description.

- Subscription

  After the above preparations have been made, the user of the product or application can be notified of the different event occurrences by subscribing to these events. Subscriptions can be created either with the aid of the TU program Notification Resources Manager or with a specific interface provided by the product or application. In Spool & Print, the parameter NOTIFICATION has been introduced in the PRINT-DOCUMENT command to provide such a specific interface.
  See section “Subscription” on page 11 for a more detailed description of subscriptions and chapter “Definition of notification resources - example” on page 27 for an example.

- Event notification

  When an event occurs, SNS generates an event notification that fully describes the event (what the event was, where it occurred, when it occurred, etc.). SNS delivers the event notifications to all the notification recipients that are subscribed to that event, if any. The event notification is delivered to the address of the notification recipient by using the notification delivery method defined in the subscription. However, an event notification is sent only if there is a corresponding subscription.

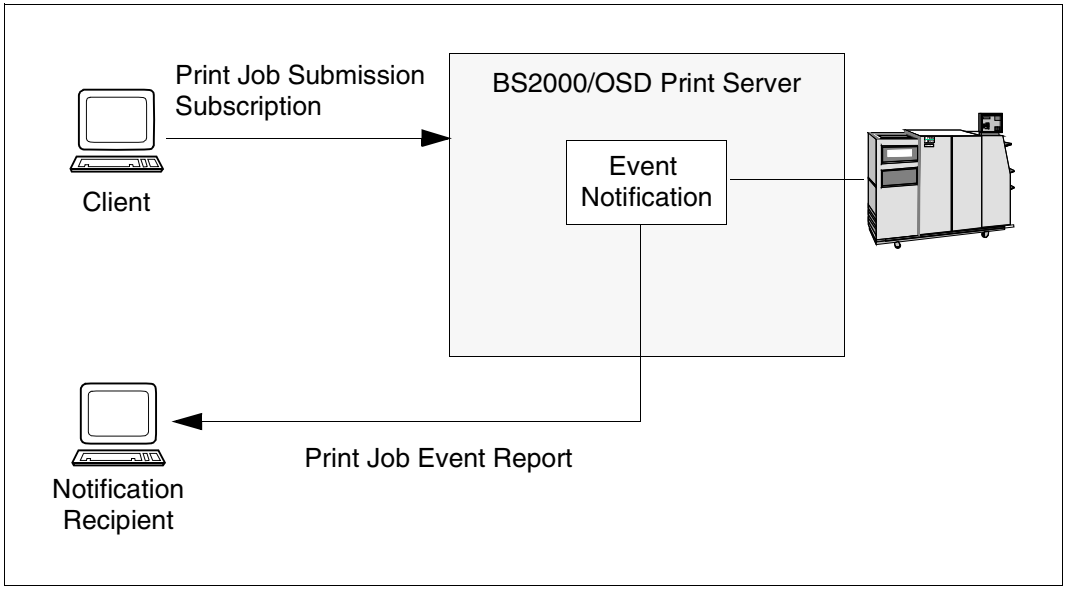

Figure 2: Event notification of a BS2000/OSD print server

2.2 Notification resources

In SNS, the following resources exist:

– object class
– event
– notification delivery method
– subscription

All resources are stored in one central file, the notification resources file NOTIFICATION.PARAMETERS, and are managed by the Notification Resources Manager program.

2.2.1 Object class

These are the objects which a user can be notified of. Events can be defined for these objects. In the frame of Spool & Print products, the object classes SPOOLJOB, RSOJOB and DPRNTJOB have been defined.

2.2.2 Event

An event defines a change of state (either expected or unexpected) of an occurrence of an object class.

An event always refers to an object class. In the frame of Spool & Print products, events such as PRINTJOBACCEPTED, PRINTJOBCOMPLETED or PRINTJOBABORTED have been defined for the object classes.

A specific API (SNPREV) is provided with SNS to allow products or applications to raise their own events i.e. to warn SNS of the occurrence of their own events. See chapter “Notification Service APIs” on page 123 for details.

2.2.3 Notification delivery method

This is the mechanism by which SNS delivers the event notifications. SNS sees the delivery methods as separate components that are plugged in SNS.

SNS V1.0B is released with different notification delivery methods (MAILTO, OPGMAIL, PROCEDURE, FILE). The components for these notification delivery methods are shipped in the library SYSLIB.SNRTP.010.METHOD.DUMMY. For a description of the notification delivery methods see the chapter “Notification delivery methods” on page 41.

2.2.4 Subscription

A subscription is a request done by a user to SNS in order to be notified when some events occurred. This user is called the subscriber.
When an event occurs, the subscription specifies to SNS how to send event notifications, where to send them and what to put into them. In the subscription four parameters must be specified:

– **When**: the subscriber has to specify the events of a specific object class he/she wants to be notified of. He/she may also specify that he/she wants to be notified of all the events of an object class.

– **How**: the subscriber has to specify the delivery method that must be used for the notification delivery.

– **Where**: the subscriber has to specify where the notification must be sent to (recipient address, procedure or file).

– **What**: the subscriber may specify which additional information he/she wants to find in his/her notification.

Thus, the subscription is an object containing a set of attributes that indicate:

– the events that cause SNS to send an event notification
– the delivery method
– the notification recipient, procedure or file
– and the information to send in an event notification.

Permanent and temporary subscriptions

Two types of subscriptions exist:

– the permanent subscriptions addressing all the occurrences of an object class.
Permanent subscriptions are registered with the Notification Resources Manager and are stored in the notification resources file NOTIFICATION.PARAMETERS.

– the temporary subscriptions addressing a single occurrence of an object class.
This occurrence is identified by an object ID whose format completely depends on the product or application raising the events. Temporary subscriptions are registered with a specific interface if such an interface is provided by the product or application raising the events. This is the case for Spool & Print that provides a NOTIFICATION parameter in its PRINT-DOCUMENT command.
A temporary subscription can be automatically removed at the end of the life of the corresponding object instance provided that a terminal event occurred. It is necessary that at least one event relative to the object class has been defined with the property TYPE=*TERMINAL.

Rules for matching of subscribed events

When an event occurs for an object class instance, SNS is informed by the API SNPREV (see section “SNPREV - Raise event interface” on page 136). SNS must then find each subscription that must be notified of this event occurrence. A subscription matches if one of the following conditions is fulfilled:

– it subscribed permanently to all the events relative to the object class the raised event belongs to

– it subscribed temporarily to the event relative to the object class the raised event belongs to

SNS will send as many notifications as there are matching subscriptions.

Privilege rules

Privilege rules for who can be notified of an event must also be respected. SNS relies on two privilege policies, a default policy and a product dedicated policy, see section “Privileges” on page 16.

2.3 Components of SNS

SNS consists of the following components:

- Notification Resources Manager
  The Notification Resources Manager (SNRMGR) is a TU program including dedicated functions for the handling of notification resources and the customisation of the notification system definition.
- Notification Resources File
  All notification resources i.e object class, event, delivery method, and subscription resources are stored in the notification resources file NOTIFICATION.PARAMETERS. The implementation of one notification delivery method (e.g. MAILTO) represents an additional and independent component to be plugged in the product.
- SNS subsystem
  The SNS subsystem consists of a TPR kernel and a TU kernel.

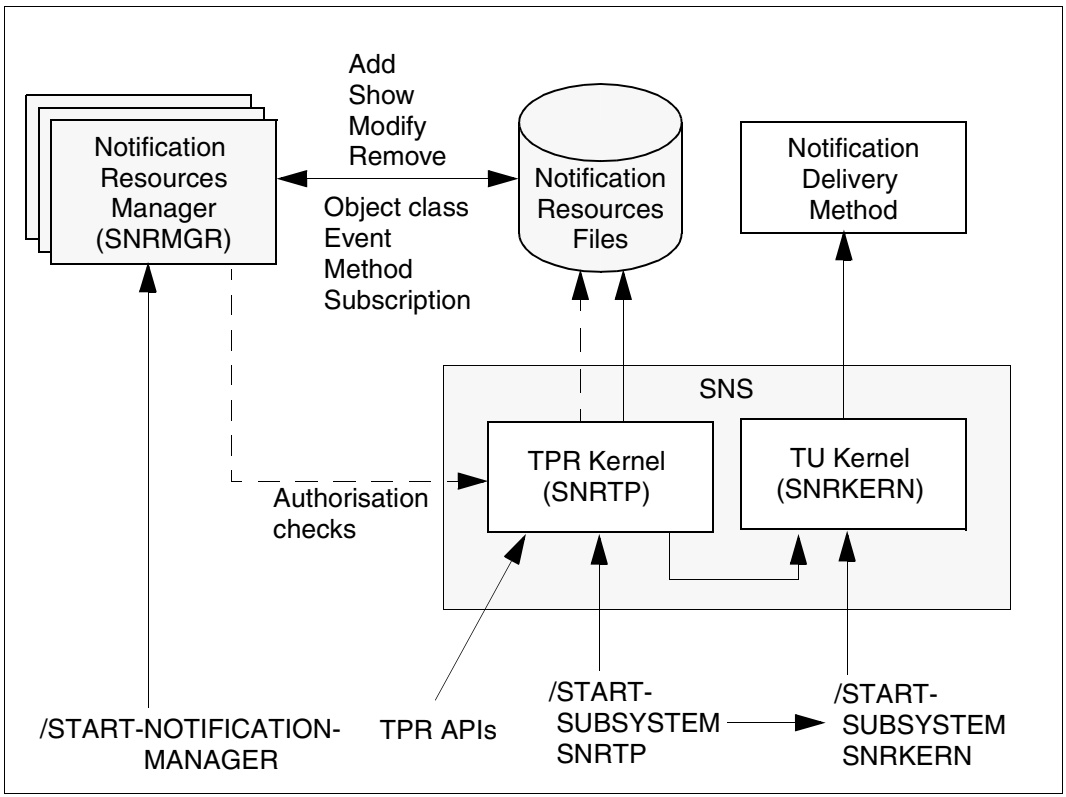

Figure 3: Components of SNS

2.3.1 Notification Resources Manager

The Notification Resources Manager (SNRMGR) manages the notification resources as objects. An object is distinguished by certain attributes. An object type is the generic term for all objects of the same type, which only differ in the values given to certain of their attributes.

The Notification Resources Manager supports the following object types:

– object class
– event
– delivery method
– subscription

All resources are stored in one central file, the notification resources file (NOTIFICATION.PARAMETERS), irrespective of their type.

The Notification Resources Manager is started with the START-NOTIFICATION-MANAGER command and is available for administrators and nonprivileged users. Some statements, however, are reserved for privileged users.

2.3.2 Notification resources file

All the notification resources i.e object olass, event, method, and subscription resources are stored in one central file. Event notification and raised event are not saved. This file MUST be saved under a specific user ID $SYSSNS that MUST NOT be located on a share PVS. After the correct loading of the subsystem, a copy of the notification resources file is also saved. If some problems occur during a session and if the notification resources file is corrupted, a restart from the copy is possible provided that the corrupted file has been erased or renamed.

Naming conventions for the notification resources file:

$SYSSNS.NOTIFICATION.PARAMETERS
$SYSSNS.NOTIFICATION.PARAMETERS.COPY

2.3.3 Notification Service TPR kernel

The Notification Service TPR kernel consists of the subsystem SNRTP and has the following responsibilities:

– management of the Notification Service subsystems

– secured access to the notification resources file

– Notification Service TPR APIs

– tuning of this kernel with the SYSSSI file

2.3.4 Notification Service TU kernel

The Notification Service TU kernel consists of a subsystem SNRKERN and has the following responsibilities:

– match the raised events to the subscriptions

– create the event notifications

– trigger the corresponding methods

2.4 Privileges

The privilege aspects are considered at two levels, The Notification Resources Manager program level and the event notification level.

Notification Resources Manager program privileges

The Notification Resources Manager program may be used by any user. However, the available functionality depends on the privileges of these users.

A privileged user, i.e. TSOS or any user having the SYSTEM-ADMINSTRATOR privilege may use the full functionality. From SECOS V4.0B, the privilege NOTIFICATION-ADMINISTRATION is available allowing this full functionality. It is necessary to have such privileges to manage object classes, events and methods.

All other users may only record subscriptions, visualize their own subscriptions and the global configuration objects.

Event notification privileges

The event notification delivery relies on two privilege policies, a default policy and a product dedicated policy.

- Default policy:

  If the owner of a subcription is TSOS or has the SYSTEM-ADMINSTRATOR privilege, he/she can be notified of any events raised for object instances of other users. Otherwise, the owner of a subscription is only allowed to be notified of events raised for his/her own object instances.

- Product policy

  The product dedicated policy is a policy provided by the product or application raising the events in the form of a module to be plugged in SNS. This policy decides if the subscription owner has the privilege to be notified or not.
  For Spool & Print, a specific privilege policy is provided. Rules for implementing and plugging in product dedicated policy modules are detailed in section “Privilege policy (optional)” on page 39.

2.5 Messages

All messages concerning SNS are classified according to the following conventions:

– The message class ID 'SNR' is reserved for all messages sent by the Notification Resources Manager program

– The message class ID 'SNK' is reserved for all messages sent by the TU kernel subsystem

– The message class ID 'SNP' is reserved for all messages sent by the TPR subsystem

– The message class IDs 'SNC', 'SNF', 'SNM' and 'SOM' are reserved for all messages sent by the respective notification delivery methods PROCEDURE, FILE, MAILTO and OPGMAIL

2.6 Tasking

Notification Resources Manager

The Notification Resources Manager program runs under the control of the dialog or batch task that launched its execution.

Notification Service

The Notification Service runs under the control of two separate processes:

– A system task SNPG controls the accesses to the NOTIFICATION.PARAMETERS files. It is this task that adds, modifies, or removes the resources in the resources file. This task also receives the events coming from the different products or applications raising their own events. Those events are dispatched to a pool of tasks (batch TU) for the notification processing. The dispatching rule is that all the events relative to the same object ID are always processed by the same task.

– A pool of batch tasks that are in charge of searching among the registered subscriptions those ones that require a notification for the just received (event,object name) duplet. The maximum number of tasks in parallel is 32. The default value is 4, but it can be tuned through the SYSSSI parameter file. Those tasks are running under the TSOS user ID.

Delivery methods

The method component is loaded in the task environment of the batch TU task. This component is totally responsible of its own tasking.

3 Installation and configuration

This section describes the software requirements needed for SNS and what must be considered during the installation and configuration of SNS.

3.1 Software requirements

The minimum software requirements for BS2000 when using SNS are as follows:

– BS2000/OSD-BC as of V4.0 (for S servers, /390 architecture)
– OSD-SVP as of V4.0 (for SR2000, RISC architecture)
– OSD/XC as of V1.0 (for SX servers, SPARC architecture)
– SDF as of V4.1

Optional software products for BS2000:

– LMS

To use the privilege NOTIFICATION-ADMINISTRATION:

– SECOS as of V4.0B

Specific customers' products for the notification delivery methods:

– *inter*Net Value Edition as of V1.0B (MAILTO)
– OPG-MAIL as of V3.2 (OPGMAIL, by OPG Online-Programmierung GmbH)

3.2 Installation

This section describes what must be considered during the installation of SNS.

Privileges necessary for the installation of SNS

SNS can be installed and managed by the following users:

– TSOS user ID
– users with the privilege SYSTEM ADMINISTRATION
– users with the privilege NOTIFICATION ADMINISTRATION
(available as of SECOS V4.0B)

Installation of SNS

SNS complies with the IMON installation conventions and is exclusively installed with IMON. The necessary inputs and the sequence of installation are described in the manual "IMON (BS2000/OSD)".

After installation, the SNS subsystem SNRTP is available, as well as the Notification Resources Manager tool which can be started with the START-NOTIFICATION-MANAGER command. This tool allows you to manipulate the resources used by the Notification Service.

The following release items are delivered for SNS:

File	Contents
SIPLIB.SNRTP.010	contains the macros
SYSFGM.SNRTP.010.E SYSFGM.SNRTP.010.D	contain the release notice of the SNS (English and German versions)
SYSLIB.SNRTP.010	contains the user API's
SYSLIB.SNRTP.010.METHOD.DUMMY	contains all the delivery method components. Yields the $SYSSNS.SYSLIB.SNRTP.METHOD library after installation.
SYSLIB.SNRTP.010.PRIV (S servers) SRMLIB.SNRTP.010.PRIV (SR2000) SPMLIB.SNRTP.010.PRIV (SX servers)	are the hardware-dependent system libraries containing the product-dedicated privilege policy components
SYSLNK.SNRTP.010 (S servers) SRMLNK.SNRTP.010 (SR2000) SPMLNK.SNRTP.010 (SX servers)	are the hardware-dependent system libraries containing the modules for the SNS TPR subsystem
SYSLNK.SNRTP.010.TU	contains the modules for the SNS TU subsystem
SYSMES.SNRTP.010	contains the message file
SYSNRF.SNRTP.010	contains the NO REF symbols

File	Contents
SYSPRC.SNRTP.010.UPDLIB	contains the procedure to convert the SYSLIB.SNRTP.010.METHOD.DUMMY library to the $SYSSNS.SYSLIB.SNRTP.METHOD library during installation
SYSRMS.SNRTP.010	contains the RMS revision package
SYSSDF.SNRTP.010	contains the commands and statement descriptions
SYSSSC.SNRTP.010	contains the subsystem catalog
SYSSPR.SNRTP.010	contains the CHANGE-FILE-NOTIFICATION command (FILE notification method)
SYSSII.SNRTP.010	contains the IMON installation information
SYSSSI.SNRTP.010	contains the tuning options

Definition of the user ID SYSSNS

This user ID must be defined on the home pubset before installing SNS. It is the repository for the notification resources file.

Starting SNS

You can now start the SNRTP subsystem with the START-SUBSYSTEM SNRTP command.
See section “Starting and Stopping SNS” on page 23.
Note that this command should be included in your system CMD file before starting any Spool & Print subsystems.

3.3 System tuning

The file SYSSSI.SNRTP.010 allows some tuning of SNS.

```
* * SYSSSI - SNRTP V01.0B
*
* SOURCE IMPLEMENTATION
* FUNKTIONS-NAME: BYPASS-TIME-STAMP-UPDATE
*
* The parameter BYPASS-TIME-STAMP-UPDATE allows to bypass the update
* of the time stamp in the notification.parameters records.
* This may be done for performance reasons.
* The possible values are:
*   BYPASS-TIME-STAMP-UPDATE=*NO (initial value): no bypass will occur
*         (information in the notification records will be updated)
*   BYPASS-TIME-STAMP-UPDATE=*YES : bypass will occur (information
*         in the notification records will not be updated)
*
BYPASS-TIME-STAMP-UPDATE=*NO
*
*
* SOURCE IMPLEMENTATION
* FUNKTIONS-NAME: SNPG-MAX-TU-TASKS-NUMBER
*
* The parameter SNPG-MAX-TU-TASKS-NUMBER specifies the maximum number of
* working tasks in charge of processing the raised events and sending
* the appropriate notifications.  The number must be a value included
* between 1 and 32
SNPG-MAX-TU-TASKS-NUMBER=4
*
*
* SOURCE IMPLEMENTATION
* FUNKTIONS-NAME: SNPG-TU-TASKS-TIME_OUT
*
* The parameter SNPG-TU-TASKS-TIME_OUT specifies the number of
* minutes the user tasks can stay without working. when the time is
* reached, the task is stopped and will be restarted when a new request
* occurs. the number of minutes must be a value included between 0
* and 120. The value 0 means that the tu tasks will stay active until
* a stop of SNRTP subsystem is detected.
SNPG-TU-TASKS-TIME_OUT=10
```

3.4 Starting and Stopping SNS

The SNS subsystem is composed of two parts: a TPR-part, named SNRTP and a TU-part, named SNRKERN.

Starting SNS

The SNRTP subsystem must be started explicitly with a START-SUBSYSTEM SNRTP command. See section “START-SUBSYSTEM - Starting the SNRTP subsystem” on page 85.

As SNS may be used by any product or application, this START-SUBSYSTEM command should be launched soon as possible after SYSTEM READY. It should be included in the CMD file as the first started subsystem or before the start of the first subsystem that uses SNS.

During the startup phase of SNRTP, the TU subsystem SNRKERN is started implicitly. This also loads the code of the Notification Resources Manager program in privileged class.

Stopping SNS

In order to ensure a continuity of SNS, the STOP-SUBSYSTEM SNRTP command should not be given during a session. This command causes the loss of events and their resulting notifications.

However, if the stop is necessary for any technical reasons, a STOP-SUBSYSTEM SNRTP command followed eventually by a STOP-SUBSYSTEM SNRTP,FORCE=YES can be submitted. This means, of course, that all events raised during the period the SNS subsystem is not running are **definitely lost**.

During the stop phase, the TU subsystem SNRKERN is implicitly stopped.

For a description of the STOP-SUBSYSTEM SNRTP command see section “STOP-SUBSYSTEM - Stopping the SNRTP subsystem” on page 86.

3.5 Registration of notification resources

To be operational, it is necessary to register resources in the notification resources file.

Object classes and events exclusively depend on the product or application providing the notification functionality. Methods exclusively depend on SNS since there are plug-ins for them.

3.5.1 Registration of object classes and events

In the framework of Spool & Print, a set of notification resources has been defined for SPOOL, Dprint, and RSO (see the documentation on these products for a description of these resources).

For each of these products, registration is performed by a specific registration procedure named SYSPRC.*product*.*version*.NOTIF. Before the procedure is run, check that the SNRTP subsystem is started.

You can visualize the registration with the Notification Resources Manager tool:

```
/START-NOTIFICATION-MANAGER
//SHOW-NOTIFICATION-RESOURCES
//END
```

Disabling notification resources

By default, all the object classes and events have the properties STATE=*ENABLE. This means that all objects and events will be notified. In order to avoid too many mails, you can disable some events that are less interesting for you.

Example

```
/START-NOTIFICATION-MANAGER
//MODIFY-NOTIFICATION-RESOURCE TYPE=*EVENT(NAME=PRINTJOBACCEPTED,
                                           OBJECT-CLASS-NAME=SPOOLJOB,
                                           STATE=*DISABLE)
...
```

This means that you will never be notified of the PRINTJOBACCEPTED event for local spool jobs. Repeat this for all the events you do not want to be notified of.

3.5.2 Registration of notification delivery methods

SNS V1.0B is released with different notification delivery methods (MAILTO, OPGMAIL, PROCEDURE, FILE).

First check if the specific customers' products for the notification delivery methods to be registered are installed correctly with IMON (see section “Software requirements” on page 19).

The definition of a notification delivery method is global for the Notification Service system and must be configured by the system administrator.

For a complete description of the delivery methods, see the chapter “Notification delivery methods” on page 41.

Registration

At SNS level, the programs performing the different notification delivery methods are stored in the library $SYSSNS.SYSLIB.SNRTP.METHOD.
This library contains the following elements for the different notification delivery methods:

Element	Content
MTHMAIL (type L) SYSMES.MTHMAIL (type X) MTHMAIL.CONFIG (type D) MTHMAIL-CLEANUP (type J) MTHMAIL (type J)	the MAILTO link and load module the MAILTO message file the MAILTO configuration file the MAILTO clean-up procedure the MAILTO procedure to call the *inter*Net Value Edition
OPGMAIL (type L) SYSMES.OPGMAIL (type X) OPGMAIL.CONFIG (type D) OPGMAIL-CLEANUP (type J)	the OPGMAIL link and load module the OPGMAIL message file the OPGMAIL configuration file the OPGMAIL clean-up procedure
MTHPROC (type L) SYSMES.MTHPROC (type X) MTHPROC.CONFIG (type D) MTHPROC-CLEANUP (type J)	the PROCEDURE link and load module the PROCEDURE message file the PROCEDURE configuration file the PROCEDURE clean-up procedure
MTHFILE (type L) SYSMES.MTHFILE (type X) MTHFILE.CONFIG (type D) MTHFILE-CLEANUP (type J)	the FILE link and load module the FILE message file the FILE configuration file the FILE clean-up procedure

The delivery methods to be used must first be registered in the notification resource file as a resource with the method name:

```
/START-NOTIFICATION-MANAGER
//ADD-NOTIFICATION-RESOURCES
    TYPE=*METHOD(NAME=method name,
                 TYPE=method type,
                 LOCATION=*LIBRARY-ELEMENT(LIBRARY=*STD,
                                           ELEMENT=method module),
                 TEMPLATE=...,
                 COMMENTS=...,
                 STATE=*ENABLE)
//END
```

where	*method name*	*method type*	*method module*	
	e.g. MAILTO	MAIL	MTHMAIL	for the MAILTO method
	e.g. OPGMAIL	MAIL-TO-BY-OPG	OPGMAIL	for the OPGMAIL method
	e.g. PROCMETH	BYPROCEDURE	MTHPROC	for the PROCEDURE method
	e.g. FILE	TOFILE	MTHFILE	for the FILE method

For a complete description of the statement and its operands, see the section “ADD-NOTIFICATION-RESOURCES” on page 88

Configuration

The delivery methods to be used may be configured.
See the relevant sections in the chapter “Notification delivery methods” on page 41.

Template

You may associate a template to the method definition, and of course you can define several methods using the same plug-in even if they are associated to different templates.
See the relavant sections in the chapter “Notification delivery methods” on page 41.

4 Definition of notification resources - example

This section shows the general notification mechanism for a notification delivery method of SNS as it works for Spool & Print using the MAILTO method as an example.
The other notification delivery methods of SNS V1.0B work in the same way.

It describes what the product that supports SNS must provide and what system administrator and nonprivileged user must do:

– notification resources in a Spool & Print environment

– registration of the notification resources to the notification resources file

– definition of subscriptions by a nonprivileged user

– definition of subscriptions by a Spool & Print administrator

– specification of subscriptions in the PRINT-DOCUMENT command

4.1 Notification resources in a Spool & Print environment

In the Spool & Print documentation, a section defines which object classes and events users can be notified of. For each event, a list of the available object attributes is provided.

The information below has to be considered as an example of the definition of notification resources and does not constitute the real specification for Spool & Print. For exact specifications, please refer to the Spool & Print documentation.

Object classes

PRINTJOB	for notifications on print jobs
PRINTER	for notifications on printers

Events for print jobs

PRINTJOBACCEPTED	raised when a print job has been accepted
PRINTJOBCOMPLETED	raised when a print job has been successfully completed
PRINTJOBCANCELLED	raised when a print job has been cancelled

For each print job event, the following object attributes are available:

TSN	tsn of the print job
USERID	user ID of the owner of the print job

For this example, only these three events and two attributes are considered. In reality, additional events as well as additional attributes are provided, see the Spool & Print manuals.

Events for printers

PRINTERREADY	raised when a printer is ready for printing
PRINTERSTOPPED	raised when a printer is stopped

For each printer event, the following object attributes are available:

PRINTERNAME	logical printer name

Notification delivery methods

MAILTO	notification by mail as provided by SNS (for example)
FORADM1	notification for the spool administrator
FORADM2	notification for the spool administrator

For each notification method, the following templates are available:

File	Content
$SYSSPOOL.PRINT-JOB-NOTIFICATION. TEMPL.FOR-ADM1	The print job with the tsn &&VARIABLE(TSN) belonging to the user &&VARIABLE(USERID) is successfully completed.
$SYSSPOOL.PRINT-JOB-NOTIFICATION. TEMPL.FOR-ADM2	The print job with the tsn &&VARIABLE(TSN) belonging to the user &&VARIABLE(USERID has been cancelled.

4.2 Registration of the notification resources

With the information on object classes, events, and methods, the system administrator has to register the notification resources in the notification resources file. This is done with the Notification Resources Manager as follows:

```
(1)  /START-NOTIFICATION-RESOURCES-MANAGER

(2)  //ADD-NOTIFICATION-RESOURCES TYPE=*OBJECT-CLASS(
                        NAME=PRINTJOB,
                        DOMAIN=SPPRINT,
                        STATE=*ENABLE)

(3)  //ADD-NOTIFICATION-RESOURCES TYPE=*EVENT(
                        NAME=PRINTJOBCOMPLETED,
                        OBJECT-CLASS-NAME=PRINTJOB,
                        TYPE=*TERMINAL,
                        STATE=*ENABLE)

(3)  //ADD-NOTIFICATION-RESOURCES TYPE=*EVENT(
                        NAME=PRINTJOBCANCELLED,
                        OBJECT-CLASS-NAME=PRINTJOB,
                        TYPE=*TERMINAL,
                        STATE=*ENABLE)

(3)  //ADD-NOTIFICATION-RESOURCES TYPE=*EVENT(
                        NAME=PRINTJOBACCEPTED,
                        OBJECT-CLASS-NAME=PRINTJOB,
                        TYPE=*NORMAL,
                        STATE=*ENABLE)

(2)  //ADD-NOTIFICATION-RESOURCES TYPE=*OBJECT-CLASS(
                        NAME=PRINTER,
                        DOMAIN=SPPRINT,
                        STATE=*ENABLE)

(3)  //ADD-NOTIFICATION-RESOURCES TYPE=*EVENT(
                        NAME=PRINTERREADY,
                        OBJECT-CLASS-NAME=PRINTER,
                        TYPE=*NORMAL,
                        STATE=*ENABLE)

(3)  //ADD-NOTIFICATION-RESOURCES TYPE=*EVENT(
                        NAME=PRINTERSTOPPED,
                        OBJECT-CLASS-NAME=PRINTERS,
                        TYPE=*TERMINAL,
                        STATE=*ENABLE)
```

```
(4)  //ADD-NOTIFICATION-RESOURCES TYPE=*METHOD(
                        NAME=MAILTO,
                        TYPE=MAIL,
                        LOCATION=*LIBRARY-ELEMENT(
                                  LIBRARY=*STD,
                                  ELEMENT=MTHMAIL(,
                                  VERSION=001),
                                  TYPE=*L),
                        TEMPLATE=*NONE,
                        STATE=*ENABLE)

(5)  //ADD-NOTIFICATION-RESOURCES TYPE=*METHOD(
                NAME=FORADM1,
                TYPE=MAILTO-BY-INETVALUE,
                LOCATION=*LIBRARY-ELEMENT(
                          LIBRARY=*STD,
                          ELEMENT=MTHMAIL(,
                          VERSION=001),
                          TYPE=*L),
                TEMPLATE=$SYSSPOOL.PRINT-JOB-NOTIFICATION.TEMPL.FOR-ADM1,
                STATE=*ENABLE),
                      COMMENTS='Method to be used when a job is completed'

(6)  //ADD-NOTIFICATION-RESOURCES TYPE=*METHOD(
                NAME=FORADM2,
                TYPE=MAILTO-BY-INETVALUE,
                LOCATION=*LIBRARY-ELEMENT(
                          LIBRARY=*STD,
                          ELEMENT=MTHMAIL(,
                          VERSION=001),
                TEMPLATE=$SYSSPOOL.PRINT-JOB-NOTIFICATION.TEMPL.FOR-ADM2,
                          STATE=*ENABLE),
                COMMENTS='Method to be used when a job is cancelled'

(7)  //END
```

(1) Starts the Notification Resources Manager program.

(2) Defines the object classes for which notifications are available. Note that the names specified for the object classes must be the ones described in the product documentation.

(3) Defines the events supported by the product (here SPOOL). Note that the names specified for the events must be the ones described in the product documentation.

(4) Defines a method resource allowing a standard notification.
This method relies (for example) on the MAILTO method of SNS i.e. the method by mail using the *inter*Net Value Edition product.

(5) Defines a method resource allowing a specific notification for the spool administrator. A template is defined for all the notifications using this method. We assume that the file $SYSSPOOL.PRINT-JOB-NOTIFICATION.TEMPL.FOR-ADM1 exists.
This method relies (for example) on the MAILTO method of SNS i.e. the method by mail using the *inter*Net Value Edition product.

(6) Defines a method resource allowing a specific notification for the spool administrator. A template is defined for all the notifications using this method. We assume that the file $SYSSPOOL.PRINT-JOB-NOTIFICATION.TEMPL.FOR-ADM2 exists.
This method relies (for example) on the MAILTO method of SNS i.e. the method by mail using the *inter*Net Value Edition product.

(7) Terminates the Notification Resources Manager program

After the configuration of the notification resources, the Notification Service is ready. Spool administrator and users can define their specific subscriptions.

4.3 Definition of subscriptions by a nonprivileged user

A nonprivileged user can define subscriptions as follows:

```
(1)  /START-NOTIFICATION-RESOURCES-MANAGER

(2)  //ADD-NOTIFICATION-RESOURCES TYPE=*SUBSCRIPTION(
                   OBJECT-USER=*OWN,
                   OBJECT-CLASS-NAME=PRINTJOB,
                   OBJECT-ID=*ALL,
                   OBJECT-ATTRIBUTES=*NONE,
                   EVENT-NAMES=*ALL,
                   USER-DATA=*NONE,
                   RECIPIENT=*PARAMETERS(
                              ADDRESS=aaa@xxx.be,
                              METHOD-NAME=MAILTO))

(3)  //ADD-NOTIFICATION-RESOURCES TYPE=*SUBSCRIPTION(
                   OBJECT-USER=*OWN,
                   OBJECT-CLASS-NAME=PRINTJOB,
                   OBJECT-ID=*ALL,
                   OBJECT-ATTRIBUTES=(TSN,USERID),
                   EVENT-NAMES=(PRINTJOBCANCELLED),
                   USER-DATA='Unfortunately, this job has been cancelled',
                   RECIPIENT=*PARAMETERS(
                              ADDRESS=aaa@xxx.be,
                              METHOD-NAME=MAILTO))
```

```
(4)  //ADD-NOTIFICATION-RESOURCES TYPE=*SUBSCRIPTION(
                   OBJECT-USER=XYZ,
                   OBJECT-CLASS-NAME=PRINTJOB,
                   OBJECT-ID=*ALL,
                   OBJECT-ATTRIBUTES=*NONE,
                   EVENT-NAMES=*ALL,
                   USER-DATA=*NONE,
                   RECIPIENT=*PARAMETERS(
                              ADDRESS=aaa@xxx.be,
                              METHOD-NAME=MAILTO))

(5)  //ADD-NOTIFICATION-RESOURCES TYPE=*SUBSCRIPTION(
                   OBJECT-USER=*OWN,
                   OBJECT-CLASS-NAME=PRINTER,
                   OBJECT-ID=L8APA,
                   OBJECT-ATTRIBUTES=*NONE,
                   EVENT-NAMES=*ALL,
                   USER-DATA=*NONE,
                   RECIPIENT=*PARAMETERS(
                              ADDRESS=aaa@xxx.be,
                              METHOD-NAME=MAILTO))

(6)  //END
```

(1) Starts the notification manager program

(2) Adds a subscription that notifies all the user's print jobs events by using the method MAILTO. This means that for each event raised for the user's print jobs, a mail will be sent to the address aaa@xxx.be. In this case, two mails will be sent and as there is no template associated to the method, their texts will be totally generated by the system:

```
Event 'PRINTJOBACCEPTED' for object 'ABCDOOOOD241ZE08 ' of object class
'PRINTJOB' has been raised.
```

and

```
Event 'PRINTJOBCOMPLETED' for object 'ABCDOOOOD241ZE08 ' of object class
'PRINTJOB' has been raised.
```

where 'ABCD0000D241ZE08 ' is the object ID i.e. in this case the tsn, sequence number and host nameof the print job

(3) Adds a subscription that raises a notification when any of the user's print jobs are cancelled. The notification is done by using the method MAILTO. The mail is addressed to aaa@xxx.be. As there is no template associated to the method, the text will be totally generated by the system. However, additional information will be present since the operands USER-DATA and OBJECT-ATTRIBUTES have been specified:

```
Event 'PRINTJOBCANCELLED' for object 'ABCDOOOOD241ZEO8 ' of object class
'PRINTJOB' has been raised.
TSN = '1234'
USERID = 'AAA     '
Unfortunately, this job has been cancelled
```

(4) Adds a subscription that raises a notification for all print jobs submitted by user ID XYZ. As the user is a nonprivileged user, no notification will be raised because the user does not have the required privileges.

(5) Adds a subscription that raises a notification for all printer events. As the user is a nonprivileged user, no notification will be raised because the user does not have the required privileges.

(6) Terminates the program

Pay attention that for each possible subscription a notification is raised.

4.4 Definition of subscriptions by a Spool & Print administrator

A Spool & Print administrator can define subscriptions as follows:

```
(1)  /START-NOTIFICATION-RESOURCES-MANAGER

(2)  //ADD-NOTIFICATION-RESOURCES TYPE=*SUBSCRIPTION(
                   OBJECT-USER=*ALL,
                   OBJECT-CLASS-NAME=PRINTJOB,
                   OBJECT-ID=*ALL,
                   OBJECT-ATTRIBUTES=*NONE,
                   EVENT-NAMES=(PRINTJOBCOMPLETED),
                   RECIPIENT=*PARAMETERS(
                             ADDRESS=tsos@xxx.be,
                             METHOD-NAME=FORADM1))

(3)  //ADD-NOTIFICATION-RESOURCES TYPE=*SUBSCRIPTION(
                   OBJECT-USER=AAA,
                   OBJECT-CLASS-NAME=PRINTJOB,
                   OBJECT-ID=*ALL,
                   OBJECT-ATTRIBUTES=(TSN,USERID),
                   EVENT-NAMES=(PRINTJOBCANCELLED),
                   USER-DATA='Unfortunately, this job has been cancelled',
                   RECIPIENT=*PARAMETERS(
                             ADDRESS=aaa@xxx.be,
                             METHOD-NAME=FORADM2))

(4)  //ADD-NOTIFICATION-RESOURCES TYPE=*SUBSCRIPTION(
                   OBJECT-USER=*OWN,
                   OBJECT-CLASS-NAME=PRINTER,
                   OBJECT-ID=L8APA,
                   OBJECT-ATTRIBUTES=*NONE,
                   EVENT-NAMES=(PRINTERSTOPPED),
                   USER-DATA=*NONE,
                   RECIPIENT=*PARAMETERS(
                             ADDRESS=tsos@xxx.be,
                             METHOD-NAME=MAILTO))

(5)  //END
```

(1) Starts the notification manager program.

(2) Adds a subscription that raises a notification when the print jobs of all users are completed. The notification is done by using the method FORADM1.
This means that when a print job is completed, a mail will be sent to the address tsos@xxx.be. As there is a template associated to the method, the text could be the following:

```
The print job with the tsn 'ABCD' belonging to the user 'AAA     ' is
successfully completed.
```

(3) Adds a subscription that raises a notification when the print jobs of user AAA are cancelled.The notification is done using the method FORADM2.
This means that when a print job of the user AAA is cancelled, a mail will be sent to the address aaa@xxx.be. As there is a template associated to the method, the text could be the following (additional information will be present since the operands USER-DATA and OBJECT-ATTRIBUTES have been specified):

```
The print job with the tsn 'ABCD' belonging to the user 'AAA     ' has
been cancelled.
TSN = '1234'
USERID = 'AAA     '
Unfortunately, this job has been cancelled
```

(4) Adds a subscription that raises a notification when the printer L8APA is stopped. The notification is done by using the method MAILTO.

This means that when printer L8APA is stopped, a mail will be sent to the address tsos@xxx.be. As there is no template associated to the method, the text will be totally generated by the system:

```
Event 'PRINTERSTOPPED' for object 'L8APA   ' of object class 'PRINTER' has
been raised.
```

(5) Terminates the program

4.5 Specification of subscriptions in the PRINT-DOCUMENT command

Users can specify in the PRINT-DOCUMENT command if and how they want to be notified or if they do not want to be notified at all. This subscription is only valid for the current job. If permanent subscriptions exist that belong to a user, specifying a new subscription in the PRINT-DOCUMENT command results in the registration of this new subscription in the resource file, so that additional notifications are sent. However, if a user specifies that he/she does not want any notification at all, all his/her registered subscriptions will be inhibited for the current job. For a detailed description of the PRINT-DOCUMENT command see the manual “Spool & Print - Commands (BS2000/OSD)”.

Example

```
/PRINT-DOCUMENT file,NOTIFICATION=*PARAMETERS(
                          OBJECT-ATTRIBUTES=*NONE,
                          EVENT-NAMES=*ALL,
                          USERDATA=*NONE,
                          RECIPIENT=*PARAMETERS(
                                    ADDRESS=xxx@yyy.zz,
                                    METHOD-NAME=MAILTO))
```

With this subscription, the user says that he/she wants to be notified of all events (EVENT-NAMES=*ALL) occurring to the current print job. He/she wants to be notified by the MAILTO method at the e-mail address xxx@yyy.zz.
This subscription will be suppressed at the end of the current print job.

5 Implementation of the notification support in a BS2000 product or application

This section explains how to implement the support of the Notification Service in a BS2000 product or application.

5.1 Identification of the notification resources

The first step when implementing the support of the Notification Service in a BS2000 product or application is to identify the notification resources that are needed.

Object classes

Define which object(s) you want to be notified of. When this has been done, define a name for each object class and define a domain name referencing the product.

Example

Object class SPOOLJOB in domain SPPRINT
Object class PRINTER in domain SPPRINT

Supported events

Determine the events relative to these object classes that should cause a notification (raise event) and give them a significant name.

Example

PRINTERSTARTED
PRINTERSTOPPED
PRINTERINERROR
PRINTJOBACCEPTED
PRINTJOBCOMPLETED
PRINTJOBCANCELLED

Supported object attributes

Determine the attributes of the object instances that could be relevant in a notification delivery.

Example

PRINTER.NAME
PRINTER.STATUS
PRINTER.ERROR.NBR

5.2 Implementation of the raise event processing

Anchor in the code implementation of the product or application the raise event processing that informs SNS of the occurrence of the event. A specific API SNPREV is provided for this purpose. See the SNPREV interface description in section “SNPREV - Raise event interface” on page 136 for details.

5.3 Documentation

Document the events and attributes available to the end users in the product or application manuals. The documentation of the product should contain the following information:

– the list of the event names and their precise description

– the names of the object instance attributes associated to the raised events as well as a description of their possible values. Those attribute names can be used in the creation of the subscriptions and in the method template.

5.4 Subscriptions at product interface (optional)

Determine whether it is necessary to provide the possibility of subscribing to the notification process directly through the product or application interface. The API SNPEVS is provided for creating such an interface, see section “SNPEVS - Create subscription interface” on page 124.

Example

In Spool & Print the PRINT-DOCUMENT command was extended with the operand NOTIFICATION=*YES(RECIPIENT-ADDRESS=….,DELIVERY-METHOD=…).

5.5 Privilege policy (optional)

If the default privilege policy provided by SNS is not sufficient, please contact your SNS vendor to envisage a dedicated solution.

6 Notification delivery methods

In SNS V1.0B, the different notification delivery methods (MAILTO, OPGMAIL, PROCEDURE, FILE) are shipped in the library SYSLIB.SNRTP.010.METHOD.DUMMY.

After an overview and a description of the operation principles, this chapter describes the different notification delivery methods in detail.

Registration

A notification delivery method that is to be used must be registered; see section “Registration of notification resources” on page 24.

Activation

The registered notification delivery methods are activated when a subscription requiring this method matches a raised event. For details see the relevant sections in the description of the notification delivery methods.

Recipient

The recipient of the notification information depends on the notification delivery method. The recipient is specified in the definition of a user subscription resource. For details see the relevant sections in the description of the notification delivery methods.

Structure

The structure of the event notification depends on the notification delivery method. For details see the relevant sections in the description of the notification delivery methods.

Configuration

The administrator can create a configuration file to specify control information for a notification delivery method. For details see the relevant sections in the description of the notification delivery methods.

Template

The layout and the content of the notification information may be customized by using a template for the notification delivery method. Each notification delivery method has its own template layout. When using templates, the template specific for the chosen notification delivery method **must** be used. For details see the relevant sections in the description of the notification delivery methods.

Control statements, conditions and variable names in the template files must always be in uppercase letters. There are no special naming conventions that must be observed for template files.

Templates can be defined at several places. They can be referenced in the method resource definition, in the method configuration file, or nowhere. The following figure describes the template selection rules for all notification delivery methods:

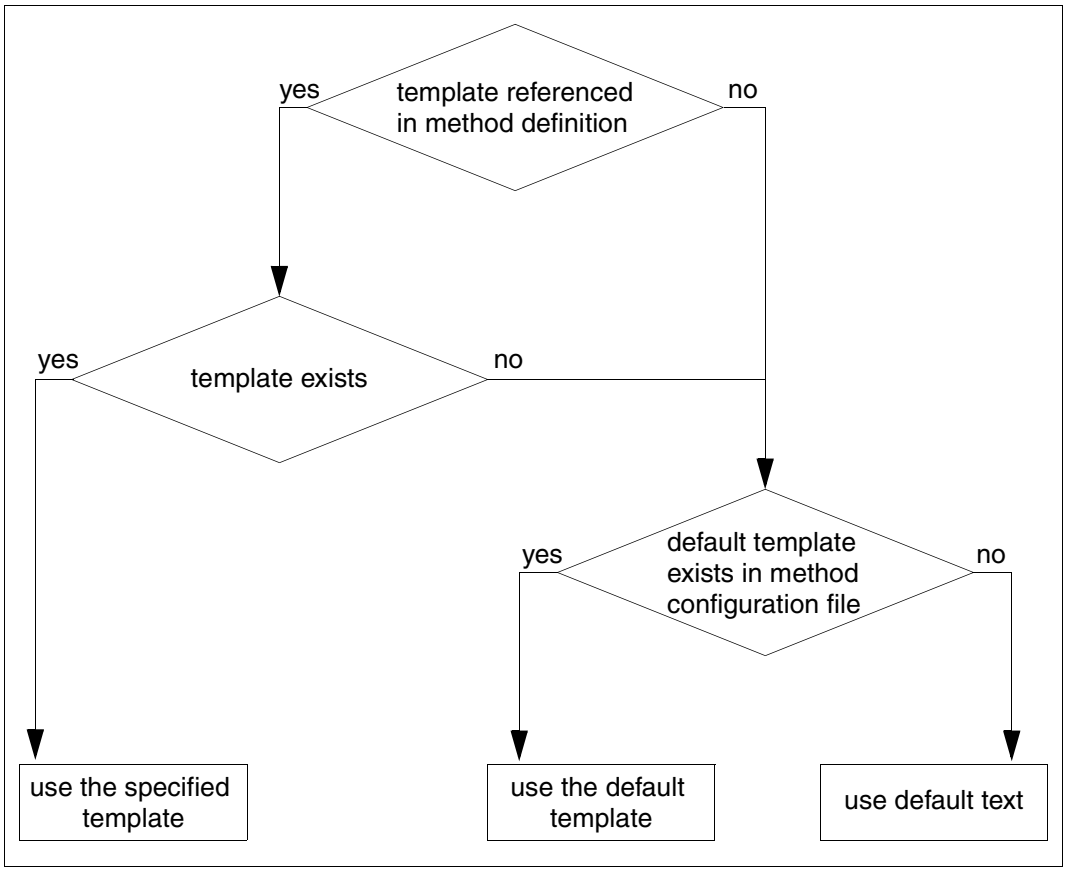

Figure 4: Template selection rules

Operation principles

The figure 5 on the next page depicts the principle of operation for all notification delivery methods. The numbers in brackets show the different operation steps. Steps (1) to (4) are common to all the notification delivery methods, whereas step (5) depends on the specific notification delivery method used.

(1) A BS2000 product (e.g. IDOM) or any application raises events

(2) Those events are sent to the SNS TPR service unit that transmits them to one of the SNS TU service tasks

(3) The SNS TU service task checks the subscriptions recorded in the notification resources file NOTIFICATION.PARAMETERS and checks if they can be applied to the event just received. For all the matching subscriptions it creates an event notification object which is identified internally by an ID, and loads the method required by the subscriptions. The method is only loaded the first time it is required.

(4) The notification method receives the event notification ID and calls SNS to get all the information relating to this ID.

(5) The MAILTO notification method prepares the mailing text (default text or according to the template) and passes this text on to the mailer agent that is in charge of sending the mail to the mail server.

The OPGMAIL notification method prepares the mail text and sends it to the recipient via OPG Mailer.

The PROCEDURE notification method prepares the parameter list for the procedure (default layout or layout according to the template), writes this parameter list to a parameter file, and (using it) starts a batch task which calls the procedure specified in the subscription.

The FILE notification method prepares the text that is to be written to the recipient file (default layout or layout according to the template) and writes it to the file.

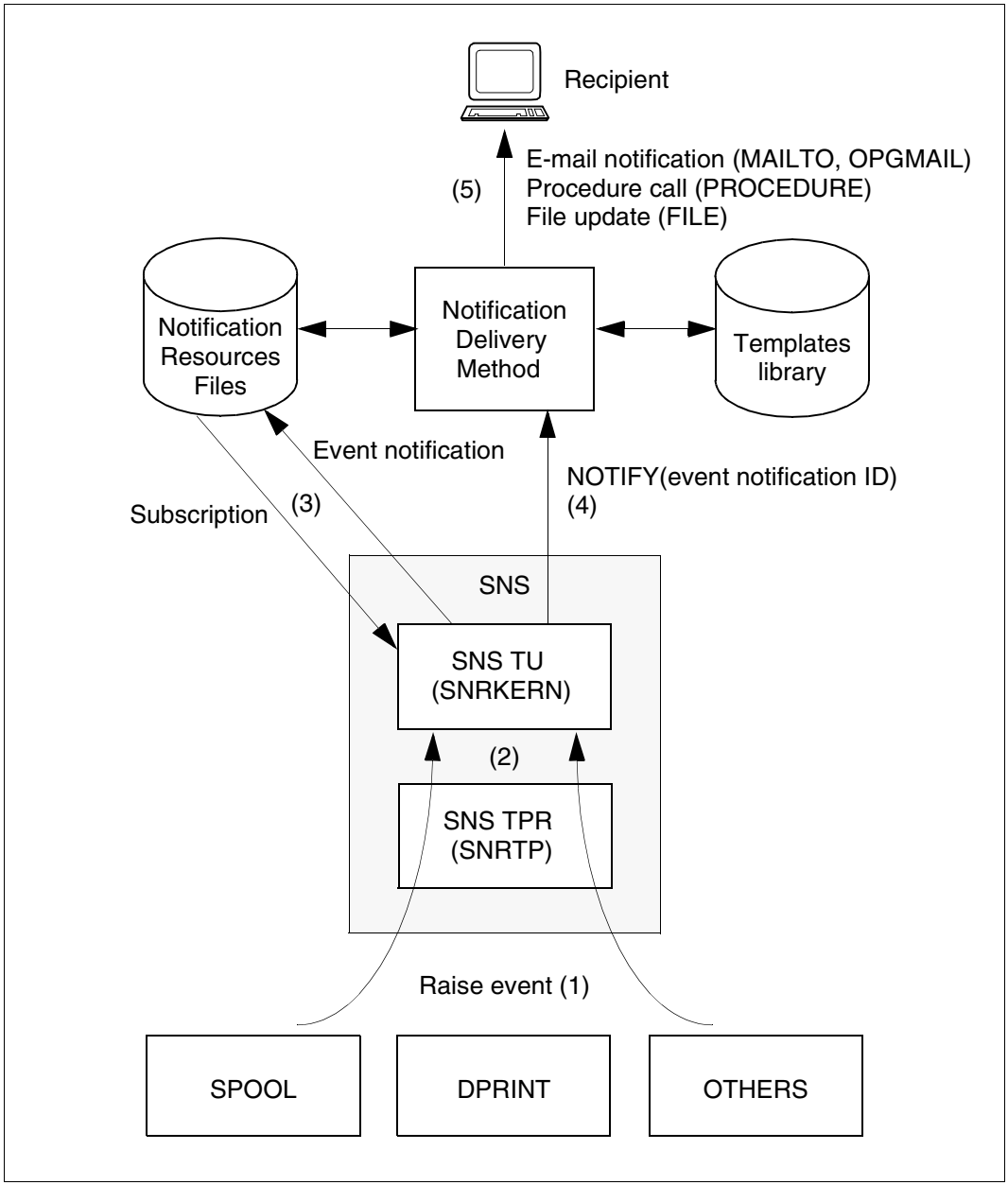

Figure 5: Principle of operation of the delivery methods

6.1 The MAILTO delivery method

This notification delivery method is responsible for sending e-mail notifications. E-mail notifications are human-readable text messages that specify the reason for the notification and the current information about the subscribed object.

Several products allow e-mails to be sent. The MAILTO notification delivery method component uses the interface with the mail services delivered with *inter*Net Value Edition as of V1.0B.
Each notification is sent to the recipient via the Simple Mail Transport Protocol (SMTP).

6.1.1 Activating the MAILTO delivery method

The MAILTO method is activated when a subscription requiring this method matches a raised event. Such a subscription can be registered in the Notification Resources Manager as follows:

```
/START-NOTIFICATION-MANAGER
//ADD-NOTIFICATION-RESOURCES
    TYPE=*SUBSCRIPTION(OBJECT-CLASS-NAME=...,
                       OBJECT-ID=...,
                       OBJECT-USER=...,
                       OBJECT-ATTRIBUTES=...,
                       EVENT-NAMES=...,
                       USER-DATA=...,
                       RECIPIENT=*PARAMETERS(ADDRESS=mail address,
                                             METHOD-NAME=method name))
```

where

mail address Mail address of the notification recipient. No check is performed upon subscription registration.

method name Logical delivery method name as defined in the Notification Resources Manager. This method must refer to the MAILTO element to activate the appropriate method.

6.1.2 MAILTO notification recipient

The recipient of the notification information is a valid e-mail address specified in the definition of a user subscription resource. This user is named “the subscription owner”.

The text layout sent depends on the template specified in the method definition or is a default layout (see section “MAILTO template” on page 52).

6.1.3 Structure of a MAILTO event notification

This section describes the content of an event notification sent via the MAILTO delivery method using the SMTP protocol.

When the Notification Service sends an e-mail message via SMTP, the content conforms to RFC 822.

E-mail messages are not guaranteed to arrive in the order they were sent, so the notification recipient may receive them in a different order.

The event notification consists of four parts: headers, message body, user data and warning area. Each part is separated by a blank line.

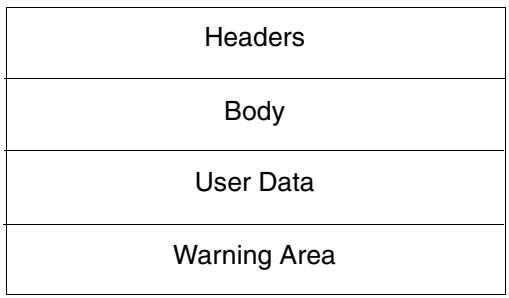

Figure 6: Structure of a MAILTO event notification

6.1.3.1 Headers

When SNS sends an event notification via SMTP, it **must** include the following headers. The header part can be omitted if the parameter `HEADER=NO` is specified in the configuration file (see section “Configuration of the MAILTO delivery method” on page 50).

“Date” header

Syntax: `Date : ` *date-time*

This header contains the date and time when the event occurred.

“From” header

Syntax: `From : ` *mailbox*

where *mailbox = addr-spec / phrase route-addr*

This header causes a typical e-mail reader to show the e-mail as coming from the notification service that is sending the event notification.
The notification service uses the second alternative for the *mailbox* syntax. The *phrase* is the name of the notification service . The *route-addr* contains an e-mail address (inside angle brackets) belonging to an administrator. This e-mail address need not be capable of receiving mail.

“Subject” header

Syntax: `Subject : *<text>`

This header specifies the subject of the message and contains a short summary of the event notification.
For job events, the `*<text>` starts with the “object name” followed by the “object ID” and then followed by the “event name” and suffixed by “completed”.

“Sender” header

Syntax: `Sender : ` *mailbox*

This header causes a typical e-mail reader to show the e-mail as coming on behalf of the person associated with the subscribing client.
If the subscription object contains the “notify-user-data” attribute, that value satisfies the RFC 822 syntax rules for *mailbox*, a “Sender” header is included, and the *mailbox* is the value of the “notify-user-data” subscription object attribute. Otherwise, no “Sender” header is included.

“Reply-to” header

Syntax: `Reply-to :` *mailbox*

If the notification recipient replies to an event notification e-mail, this header causes a typical e-mail reader to send an e-mail to the person acting as the subscribing client. The rules are identical to the “Sender” header.

“To” header

Syntax: `To :` *mailbox*

This header specifies the notification recipient(s).
This header must be included. The *mailbox* must be the address value of the NOTIFY-RECIPIENT subscription attribute.

“Content-type” header

Syntax: `Content-Type :` *type*`/`*subtype* `*(;`*parameter*`)`

This header specifies the format of the message body.
The Notification Service must include the “Content-Type” header. The possible value is the `Text/Plain` value. See RFC 1521 for the syntactic terms (e.g. *type*)

6.1.3.2 Body

The message body of an event notification must contain human-consumable content as plain text. If a template exists, the message body contains the template text with variables and rules converted.

If no template exists, the body contains this default information :

```
Event "event name" for object "object ID" of object class "object name" has
been registered at "date time".
```

Example

For SPOOL print job objects a default information could be:

```
Event PRINTJOBACCEPTED for object ABCDOOOOD241ZE2O of object class SPOOLJOB
has been registered at 2004-08-16 16:23:40
```

6.1.3.3 User Data

The user data part of an event notification is composed of two sections:

– the notification attribute part, comprising the values of the attributes specified in the subscription resource object.
– the user data part, comprising the user data value specified in the subscription resource object.

These two parts are separated by a blank line.

Example

The following subscription was added in the notification resources file:

```
/START-NOTIFICATION-MANAGER
//ADD-NOTIFICATION-RESOURCES
    TYPE=*SUBSCRIPTION(OBJECT-CLASS-NAME=SPOOLJOB,
                       OBJECT-USER=TOTO,
                       OBJECT-ATTRIBUTES=(TSN,USERID,ACCOUNT),
                       USER-DATA=THIS IS MY USER DATA,
                       RECIPIENT=*PARAMETERS(ADDRESS=USERB@XYZ.COM,
                                             METHOD-NAME=MAILTO))
```

The user data part of an event notification will be:

```
...
TSN = ABCD
USERID = TOTO
ACCOUNT = ACC2

THIS IS MY USER DATA
```

6.1.3.4 Warning area

The warning area of an event notification informs the mail recepient about the contents of the header and body part. Three warning messages can be present in this section:

`*** WARNING : DEFAULT HEADER IN MAIL FILE ***`
When no configuration file has been detected, the default header part is generated and this message is inserted in the warning area.

`*** WARNING : DEFAULT TEMPLATE USED IN MAIL FILE ***`
When no template file or an erroneous template file was associated with the MAILTO delivery method, the default template defined in the configuration file is used.

`*** WARNING : NO TEMPLATE USED IN MAIL FILE ***`
When no template is present, only the default information (see section “Body” on page 48) is present in the body part.

6.1.4 Configuration of the MAILTO delivery method

The MTHMAIL program must be configured. A configuration file must be created by the system administrator and must be present in the $SYSSNS.SYSLIB.SNRTP.METHOD library under the element name MTHMAIL.CONFIG of type D. MTHMAIL.CONFIG is the unique configuration file used by the method.

The following must be configured:

– whether or not a header should be present
– the contents of the header (administrator, sender and reply address)
– the template used by default if no template is specified in the method resource definition
– the address of the SMTP server (mail server)

If the configuration file is not present or not accessible, a default header part will be included in the mail text and `SMTPSERVER=*LOCAL` is used.

The following lines can be included in the configuration file:

Option	Description
`HEADER=YES | NO`	If `HEADER=YES` is present in the confguration file, the header part will be included in the mail text. If `HEADER=NO` (or any other value) is present in the configuration file, no header part will be included in the mail text.
`ADMINISTRATOR=tsos@xyz.com`	Mail address of the sender (generally the BS2000 administrator mail address). If this line is not present, the header part line in the mail text will be: `From : ** ERROR **`
`SENDER=aaa@xyz.com`	Mail address of a general notification responsible. The `SENDER` field is optional.
`REPLY=bbb@xyz.com`	Mail address of a notification responsible with which the recipient of the mail can correspond (generally the same mail address as the sender). In fact, the recipient of a notification receives a mail from a BS2000 system. `SENDER` and `REPLY` information allow him/her to have an e-mail address to communicate with the notification responsible. The `REPLY` field is optional.

Option	Description
FILENAME=*template file name*	File name of the default template file used. This template is used if there is no template associated with the MAILTO delivery method or the template does not exist or is not accessible. This default template can also be a library element. In this case the line FILENAME is replaced by these four lines : LIBRARY=*template library name* ELEMENT=*library element* TYPE=D VERSION=001
SMTPSERVER=aa.bb.cc.dd	IP address of the host or the name of the BCAM host on which the SMTP server is running and to which the mail is to be sent. The SMTPSERVER field is mandatory

Note

When the configuration file is accessed (e.g. in order to be modified), it is then locked for other users. If at this time the notification mechanism wants to access the locked configuration file, it disables the method. Corresponding information is then given at the system console.

6.1.5 MAILTO template

A MAILTO template is a SAM file that may contain:

– Variable fields, i.e. fields that reference an object attribute and that are replaced in the mail text by the current value of the object attribute. Variable fields must be defined as follows:

  `&&VARIABLE(`*variable name*`)`

  Invalid variable names are replaced by the error string `*** error ***`
  Valid variable names that are empty are replaced by the string `*not relevant`.

  The list of the keywords available for the variable name are described in the documentation of the product or the application raising the events. In the context of SPOOL, this information is available in the “SPOOL (BS2000/OSD)” documentation.

– Conditional commands that allow conditional text generation. Conditional text generation must be defined as follows:

  `&&IF(`*condition*`) ... <text> ... &&ENDIF()`

  The list of the keywords available for supported conditions are described in the documentation of the product or the application raising the events. In the context of SPOOL, this information is available in the “SPOOL (BS2000/OSD)” documentation.

– Constant text that is free text which is reproduced without any change in the mail text.

Each line of the template is interpreted and copied to the message body. The variables (if any) are substituted by their values (in string form). Texts included in the conditional text section will be interpreted and copied only if the condition is satisfied.

Example

Template text:

```
This is a template example.
&&IF(EVENT=PRINTJOBCOMPLETED)
The print job with tsn &&VARIABLE(TSN) has been successfully processed.
&&ENDIF()
&&IF(EVENT=PRINTJOBABORTED)
The print job with tsn &&VARIABLE(TSN) belonging to user
&&VARIABLE(USERID)
Has been cancelled.
&&ENDIF()
This is the end of the example.
```

6.1.6 Example of the MAILTO delivery method

This section contains an example of a mail notification for a print job event.

The following subscription is registered in the notification resources file:

```
/START-NOTIFICATION-MANAGER
//ADD-NOTIFICATION-RESOURCES
    TYPE=*SUBSCRIPTION(OBJECT-CLASS-NAME=SPOOLJOB,
                       OBJECT-USER=TOTO,
                       OBJECT-ATTRIBUTES=(TSN,USERID,ACCOUNT),
                       USER-DATA=THIS IS MY USER DATA,
                       RECIPIENT=*PARAMETERS(ADDRESS=USERB@XYZ.COM,
                                             METHOD-NAME=MAILTO))
```

The following template is associated with the MAILTO delivery method:

```
This is a template example.
&&IF(EVENT=PRINTJOBCOMPLETED)
The print job with tsn &&VARIABLE(TSN) has been successfully processed.
&&ENDIF()
&&IF(EVENT=PRINTJOBABORTED)
The print job with tsn &&VARIABLE(TSN) belonging to user
&&VARIABLE(USERID)
Has been cancelled.
&&ENDIF()
This is the end of the example.
```

When the print job is completed, SNS generates and sends the following e-mail message:

```
Date : 2004-08-16 / 16:23:40
From : TSOS@XYZ.COM
Subject : SPOOLJOB ABCD0000D241ZE08 PRINTJOBCOMPLETED completed.
Sender: USERA@XYZ.COM
Reply-to: USERA@XYZ.COM
To : USERB@XYZ.COM
Content-Type : Text/Plain

This is a template example.
The print job with tsn ABCD has been successfully processed.
This is the end of the example.

TSN = ABCD
USERID = TOTO
ACCOUNT = OSL

THIS IS MY USER DATA ...
```

6.2 The OPGMAIL delivery method

This notification delivery method is responsible for sending e-mail notifications. E-mail notifications are human-readable text messages that specify the reason for the notification and the current information about the subscribed object.

Each notification is sent to the recipient via the Simple Mail Transport Protocol (SMTP). The MAIL product of the OPG company is used for mailing. For more details about the MAIL product, please refer to the internet (URL: *http://www.opg.de*). There you will also find the manual "OPG MAIL für BS2000" (in German only).

When an event occurs, the Notification Service gathers all the subscriptions interested in this event and addresses the notification information on the OPGMAIL delivery method which builds an e-mail text and sends it to the recipient.

6.2.1 Activating the OPGMAIL delivery method

The OPGMAIL method is activated when a subscription requiring this method matches a raised event. Such a subscription can be registered in the Notification Resources Manager as follows:

```
/START-NOTIFICATION-MANAGER
//ADD-NOTIFICATION-RESOURCES
    TYPE=*SUBSCRIPTION(OBJECT-CLASS-NAME=...,
                       OBJECT-ID=...,
                       OBJECT-USER=...,
                       OBJECT-ATTRIBUTES=...,
                       EVENT-NAMES=...,
                       USER-DATA=...,
                       RECIPIENT=*PARAMETERS(ADDRESS=mail address,
                                             METHOD-NAME=method name))
```

where

mail address Mail address of the notification recipient. No check is performed upon subscription registration.

method name Logical delivery method name as defined in the Notification Resources Manager. This method must refer to the OPGMAIL element to activate the appropriate method.

6.2.2 OPGMAIL notification recipient

The recipient of the notification information is a valid e-mail address specified in the definition of a user subscription resource. This user is named “the subscription owner”.

The text layout sent depends on the template specified in the method definition or is a default layout (see section “OPGMAIL template” on page 61).

6.2.3 Structure of an OPGMAIL event notification

This section describes the content of an event notification sent via the OPGMAIL delivery method using the SMTP protocol.

When the Notification Service sends an e-mail message via SMTP, the content conforms to RFC 822.

E-mail messages are not guaranteed to arrive in the order they were sent, so the notification recipient may receive them in a different order.

The event notification consists of four parts: headers, message body, user data, and warning area. The headers precede the message body, and each part is separated by a blank line.

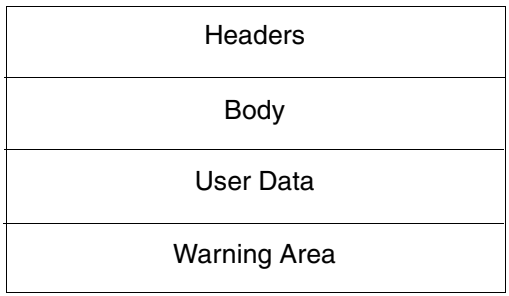

Figure 7: Structure of an OPGMAIL event notification

6.2.3.1 Headers

When SNS sends an event notification via SMTP, it **must** include the following headers. RFC 822 **recommends** that the headers be in the order that they appear below.

The header part can be omitted if the parameter `HEADER=NO` is specified in the configuration file (see section “Configuration of the OPGMAIL delivery method” on page 59).

“Date” header

Syntax: `Date : ` *date / time*

This header contains the date and time when the event occurred.

“From” header

Syntax: `From : ` *mailbox*

where *mailbox = addr-spec / phrase route-addr*

This header causes a typical e-mail reader to show the e-mail as coming from the notification service that is sending the event notification.
The notification service uses the second alternative for the *mailbox* syntax. The *phrase* is the name of the notification service . The *route-addr* contains an e-mail address (inside angle brackets) belonging to an administrator. This e-mail address need not be capable of receiving mail.

“Subject” header

Syntax: `Subject : *<text>`

This header specifies the subject of the message and contains a short summary of the event notification.
For job events, the `*<text>` starts with the “object name” followed by the “object identification” and then followed by the “event name”.

“Sender” header

Syntax: `Sender : ` *mailbox*

This header causes a typical e-mail reader to show the e-mail as coming on behalf of the person associated with the subscribing client.
If the subscription object contains the “notify-user-data” attribute, that value satisfies the RFC 822 syntax rules for *mailbox*, a “Sender” header is included, and the *mailbox* is the value of the “notify-user-data” subscription object attribute. Otherwise, no “Sender” header is included.

“Reply-to” header

Syntax: `Reply-to :` *mailbox*

If the notification recipient replies to an event notification e-mail, this header causes a typical e-mail reader to send an e-mail to the person acting as the subscribing client. The rules are identical to the “Sender” header.

“CC” header

Syntax: `CC :` *mailbox*

This header defines the recipients of the mail who are to receive a copy of the mail. The rules are identical to the “Sender” header

“To” header

Syntax: `To :` *mailbox*

This header specifies the notification recipient(s).
This header must be included. The *mailbox* must be the address value of the NOTIFY-RECIPIENT subscription attribute.

“Content type” header

Syntax: `Content-Type :` *type/subtype* `*(;`*parameter*`)`

This header specifies the format of the message body.
The Notification Service must include the ”Content type“ header. The possible value is `Text/Plain` value. See RFC 1521 for the syntactic terms (e.g. *type*).

6.2.3.2 Body

The message body of an event notification must contain human-consumable content as plain text. If a template exists, the message body contains the template text with variables and rules converted.

If no template exists, the body contains this default information :

```
Event "event name" for object "object ID" of object class "object name" has
been registered at "date time".
```

6.2.3.3 User data

The user data part of an event notification is composed of two sections:

– the notification attribute part, comprising the values of the attributes specified in the subscription resource object

Example

```
...
TSN = ABCD
USERID = USER
ACCOUNT = ACC2
```

– the user data part, comprising the user data value selected via the PRINT-DOCUMENT command

These two parts are separated by a blank line.

6.2.3.4 Warning area

The warning area informs the mail recepient about the content of the header and body parts.
Three warning messages and two error messages can be present in this section:

`*** WARNING : DEFAULT HEADER IN MAIL FILE ***`
If no configuration file has been detected, the default header part is generated and this message is inserted in the warning area.

`*** WARNING : CONFIGURATION TEMPLATE USED IN MAIL FILE ***`
As no template file or an erroneous template file is associated with the OPGMAIL method, the default template defined in the configuration file is used.

`*** WARNING : NO TEMPLATE USED IN MAIL FILE ***`
As no template file or an erroneous template file is associated with the OPGMAIL method or no template or an erroneous default template is defined in the configuration file, only the default information (see section “Body” on page 57) is present in the body part as no template is present.

`*** ERROR : DMS ERROR DURING TEMPLATE TREATMENT ***`
As a DMS error occurs during template treatment, only the default information (see section “Body” on page 57) is present in the body part as no template is present.

`*** ERROR : TEMPLATE TREATMENT NOT CORRECT ***`
As the template's variable fields and conditional commands error appear, this message is displayed in this section (see section “Body” on page 57).

6.2.4 Configuration of the OPGMAIL delivery method

The administrator can create a configuration file to specify some parameters for OPGMAIL.

The file must be present in the $SYSSNS.SYSLIB.SNRTP.METHOD library under the element name OPGMAIL.CONFIG of type D.

The following lines can be included in the configuration file:

Option	Description
HEADER=YES | NO	If HEADER=YES is present in the confguration file, the header part will be included in the mail text. If HEADER=NO (or any other value) is present in the configuration file, no header part will be included in the mail text.
ADMINISTRATOR=tsos@xyz.com	Mail address of the sender (here generally the BS2000 administrator mail address). This field contains the identity of the person who wished this message to be sent. If this line is not present, the header part line in the mail text will be: From : ** ERROR **
SENDER=aaa@xyz.com	Mail address of a general notification responsible. This field contains the authenticated identity of the agent (person, system or process) that sends the message. The corresponding parameter of OPG Mail is FROM.
CC=bbb@xyz.com	Mail address to receive a copy of the mail
BCC=ccc@xyz.com	Mail address to receive a blind copy of the mail
RR=ddd@xyz.com	Mail address to receive a receipt of the mail
REPLY=eee@xyz.com	Mail address of a notification responsible with which the recepient of the mail can correspond (generally the same mail address as the sender).
DOMAIN=*domain name*	Internet domain used. In the case of address mail given without domain (in all fields with mail address), this domain is added as the default domain.
DOMAINADD=YES | NO	Specifies whether or not the DOMAIN field is used.
MAILERASE=YES | NO	In the case of error or warning while the mail is being sent via the OPG Mailer, this parameter specifies whether or not the mail file (OPGMAIL.TXT.*notid*) is erased. By default the mail file is not erased.

Option	Description
FILENAME=*template file name*	File name of the default template file used. This template is used if there is no template associated with the mail method or the template does not exist or is not accessible. This default template must also be a library element. In this case the line FILENAME is replaced by these four lines: LIBRARY=*template library name* ELEMENT=*library element* TYPE=D VERSION=001
SERVER=*server name*	Name of the BCAM host on which the SMTP server is running and to which the mail is to be sent. If this parameter is not defined, the default server name defined in the OPG Mail general configuration file (MAIL.PAR) is used.

If the configuration file is not present or not accessible, a header default part will be included in the mail text.

As a large number of parameters (excluding HEADER, ADMINISTRATOR, MAILERASE, FILENAME, LIBRARY, ELEMENT, TYPE, VERSION) are used by OPG MAIL, see the manual “OPG MAIL für BS2000” (in German only) for more information on the parameters.

6.2.5 OPGMAIL template

An OPGMAIL template is a SAM file that may contain:

– Variable fields, i.e. fields that reference an object attribute and that are replaced in the mail text by the current value of the object attribute. Variable fields must be defined as follows:

  `&&VARIABLE(`*variable name*`)`

  Invalid variable names are replaced by the error string `*** error ***`
  Valid variable names that are empty are replaced by the string `*not relevant`.

  The list of the keywords available for the variable name are described in the documentation on the product or the application raising the events. In the context of SPOOL, this information is available in the “SPOOL (BS2000/OSD)” documentation.

– Conditional commands that allow conditional text generation. Conditional text generation must be defined as follows:

  `&&IF(`*condition*`) ... <text> ... &&ENDIF()`

  The list of keywords available for the conditions supported is described in the documentation on the product or the application raising the events. In the context of SPOOL, this information is available in the “SPOOL (BS2000/OSD)” documentation.

– Option declarations that allow global method options contained in the method configuration file to be overwritten. The option syntax is:

  ##option name=option value

  beginning in the first column of any line.

  The list of the possible options is described in section “Configuration of the OPGMAIL delivery method” on page 59.

– Constant text that is free text which is reproduced without any change in the mail text.

Each line of the template is interpreted and copied to the message body (except the option declarations lines) with the variables being substituted by their values (in string form). Texts included in the conditional text section will be interpreted and copied only if the condition is satisfied.

Example

Template text:

```
##ADMINISTRATOR=TXT2PDF
This is a template example.
&&IF(EVENT=PRINTJOBCOMPLETED)
The print job with tsn &&VARIABLE(TSN) has been successfully processed.
&&ENDIF()
&&IF(EVENT=PRINTJOBABORTED)
The print job with tsn &&VARIABLE(TSN) belonging to user &&VARIABLE(USERID)
Has been cancelled.
&&ENDIF()
This is the end of the example.
```

6.2.6 Example for the OPGMAIL delivery method

This section contains an example of a mail notification for a print job event.

The following subscription is registered in the notification resources file:

```
/START-NOTIFICATION-MANAGER
//ADD-NOTIFICATION-RESOURCES
    TYPE=*SUBSCRIPTION(OBJECT-CLASS-NAME=SPOOLJOB,
                       OBJECT-USER=TOTO,
                       OBJECT-ATTRIBUTES=(TSN,USERID,ACCOUNT),
                       USER-DATA=THIS IS MY USER DATA,
                       RECIPIENT=*PARAMETERS(ADDRESS=USERB@XYZ.COM,
                                             METHOD-NAME=OPGMAIL))
```

The following configuration file OPGMAIL.CONFIG is present in the $SYSSNS.SYSLIB.SNRTP.METHOD library:

```
ADMINISTRATOR=TSOS@XYZ.COM
SENDER=USERA@XYZ.COM
REPLY=USERR@XYZ.COM
MAILERASE=YES
DOMAIN=XYZ.COM
DOMAINADD=YES
```

The following template is associated with the OPGMAIL delivery method:

```
##ADMINISTRATOR=APPL@XYZ.COM
##CC=&&VARIABLE(USERID)
This is a template example.
&&IF(EVENT=PRINTJOBCOMPLETED)
The print job with tsn &&VARIABLE(TSN) has been successfully processed.
&&ENDIF()
&&IF(EVENT=PRINTJOBABORTED)
The print job with tsn &&VARIABLE(TSN) belonging to user &&VARIABLE(USERID)
Has been cancelled.
&&ENDIF()
This is the end of the example.
```

A subscribing Client `APPL` (who works for `xyz` corp.) performs a subscription creation operation for a print job operation on printer `tiger`.

When the print job is completed, SNS generates and sends the following e-mail message:

```
Date : 2003-10-21 / 16:23:40
From : APPL@XYZ.COM
Subject : SPOOLJOB ABCE0000D241ZE08 PRINTJOBCOMPLETED
Sender : USERA@XYZ.COM
Reply-to : USERR@XYZ.COM
Cc : USERC@XYZ.COM
To : USERB@XYZ.COM
Content-Type : Text/Plain

This is a template example.
The print job with ABCD has been successfully processed.
This is the end of the example.

TSN = ABCD
USERID = USERC
ACCOUNT = OSL

THIS IS MY USER DATA

*** WARNING : DEFAULT HEADER IN MAIL FILE ***
```

In this mail you can see an example with the four sections: header, body, user data (with attribute part and user data part) and warning.

Remarks

– The `From` field of the header part contains `APPL@XYZ.COM`. This is the value of the `##ADMINISTRATOR` field of the template (that overwrites the value of the `ADMINISTRATOR` field defined in the configuration file.

– The `Cc` field contains the variable *userid* specified in the template (*userid* value is `USERC`). This field is suffixed by `@XYZ.COM` as the fields `DOMAIN=XYZ` and `DOMAINADD=YES` are present in the configuration file.

6.2.7 Diagnosis

In the case of an error or warning while a mail is being sent via the OPG Mailer, a file is created with the OPG messages returned.

The name is: `SYSOUT.OPGMAIL.TXT.`*notid* with the *notid* associated with the mail.

In this case the mail file (`OPGMAIL.TXT.`*notid*) is either erased or not depending on the `MAILERASE` parameter of the configuration or template file. By default the mail file is not erased.

6.3 The PROCEDURE delivery method

This notification delivery method is responsible for performing the notification by starting a procedure.

When an event occurs, the Notification Service gathers all the subscriptions interested in this event and addresses the notification information to the PROCEDURE delivery method that starts the procedure associated with the subscription.

The information about the event and about the object is available as input for the procedure, which is always an SDF-P procedure. The SDF-P procedure runs in batch mode and (for security reasons) under the user ID of the subscription owner.

6.3.1 Activating the PROCEDURE delivery method

The PROCEDURE method is activated when a subscription requiring this method matches a raised event. Such a subscription can be registered in the Notification Resources Manager as follows:

```
/START-NOTIFICATION-MANAGER
//ADD-NOTIFICATION-RESOURCES
    TYPE=*SUBSCRIPTION(OBJECT-CLASS-NAME=...,
                       OBJECT-ID=...,
                       OBJECT-USER=...,
                       OBJECT-ATTRIBUTES=...,
                       EVENT-NAMES=...,
                       USER-DATA=...,
                       RECIPIENT=*PARAMETERS(ADDRESS=procedure name,
                                             METHOD-NAME=method name))
```

where

procedure name Name of the SDF-P procedure to be run. No check is performed upon subscription registration. This name can be specified as a file or library element. It must comply with the FROM-FILE operand of the CALL-PROCEDURE command. The file name or the library name may contain `catid` and `userid` references, but they have to be shareable if the subscription owner is not the TSOS user ID. Violations of these rules result in an error message and no notification.

method name Logical delivery method name as defined in the Notification Resources Manager. This method must refer to the PROCEDURE element to activate the appropriate method.

6.3.2 PROCEDURE notification recipient

The recipient of the notification is an SDF-P procedure whose name is specified in a user subscription resource. This user is named “the subscription owner”.

The parameter list of this procedure is provided in a parameter file. Its contents depend on the template specified in the method definition or configuration file and on the attributes OBJECT-ATTRIBUTES and USER-DATA specified in the subscription.
See section “ADD-NOTIFICATION-RESOURCES” on page 88.

The parameter file has the file attributes 'SAM, STD(2), RECFORM=V'.

Its name is $*userid*.SNRTP.*tsn*.MTHPROC.PL.*cpu*.

where

userid	User ID of the subscription owner
tsn	Task serial number of the SNRTP TU batch task that runs the method
cpu	CPU time in 1/10000 seconds used by the batch task since its creation (to avoid name collisions)

The method creates and starts a batch job under the user ID of the subscription owner.

The name of the batch file is $*userid*.SNRTP.*tsn*.MTHPROC.BATCH.*cpu*
where *userid*, *tsn* and *cpu* are as described above.

This batch job calls the user procedure as specified in the subscription. As a parameter to the user procedure, the name of the parameter file is given.

Example

```
/CALL-PROCEDURE -
/   FROM-FILE=MY-SDF-PROGRAM, -
/   PROC-PARAM=(PL=$USER1.SNRTP.ABCD.MTHPROC.PL.XXXX)
```

or

```
/CALL-PROCEDURE -
/   FROM-FILE=*LIB(MYLIB,MYPROC), -
/   PROC-PARAM=(PL=$USER1.SNRTP.ABCD.MTHPROC.PL.XXXX)
```

Note

The procedure parameter list file $*userid*.SNRTP.*tsn*.MTHPROC.PL.*cpu*) will be deleted either immediately by the method or upon SNRTP subsystem termination by a clean-up batch job if the associated batch file is no longer present. The clean-up batch job (job name MTHPROCC) is started when the method is first called.
Synchronization of the clean-up job with the SNRTP subsystem is performed with the $TSOS.SNRTP job variable.

6.3.3 Structure of a PROCEDURE event notification

This section describes the parameter file content that is transmitted to the SDF-P procedure.

This parameter file consists of three parts: template information, object attributes, user data.

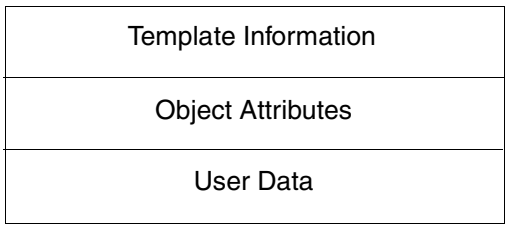

Figure 8: Structure of the PROCEDURE parameter file

6.3.3.1 Template information

If no template is associated with the method, this part contains the following default information:

```
EVENT-DATE = YYYY-MM-DD
EVENT-TIME = HH:MM:SS
EVENT-NAME = event name
OBJECT-CLASS-NAME = object class name
OBJECT-ID = object ID
```

If a template is associated with the method, this part is built according to the template layout. See the section “PROCEDURE template” on page 69 for details.

6.3.3.2 Object attributes

This part contains the attributes specified in the subscription definition in the form:

attribute name = attribute value

If `OBJECT-ATTRIBUTES=*ALL` is specified in the subscription definition (see section “ADD-NOTIFICATION-RESOURCES” on page 88), all the object attributes are presented here. If `OBJECT-ATTRIBUTES=*NONE` is specified in the subscription definition, this part is empty.

6.3.3.3 User data

This part contains the user data information as specified in the subscription definition in the form:

USER-DATA = *user data information*

If USER-DATA=*NONE is specified in the subscription definition (see section “ADD-NOTIFICATION-RESOURCES” on page 88), this part is empty.

6.3.4 Configuration of the PROCEDURE delivery method

The administrator can create a configuration file to specify some parameters for PROCEDURE.

The file must be present in the $SYSSNS.SYSLIB.SNRTP.METHOD library under the element name MTHPROC.CONFIG of type D.

The following lines can be included in the configuration file:

Option	Description
JOB-CLASS=<name 1..8>	JOB-CLASS used for starting the batch job.
ACCOUNT=*FIRST | <alphanum-name 1..8>	Account number of the user ID to be used by the batch job. ACCOUNT=*FIRST uses the first account number of the user ID.
CPU-LIMIT=*STD | *NO | *BY-JOB-CLASS | <integer 1..32767>	Maximum CPU time to be used by the batch job.
FILENAME=*template file name*	File name of the default template file used. This template is used if there is no template associated with the mail method or the template does not exist or is not accessible. This default template can also be a library element; in this case the FILENAME line is replaced by these four lines: LIBRARY=*template library name* ELEMENT=*library element* TYPE=D VERSION=001

If one of the batch job parameters is not present in the configuration file, the batch job will be started with the corresponding default parameter value (JOB-CLASSS=JCBSYS and/or ACCOUNT=*FIRST and/or CPU-LIMIT=*NO).

6.3.5 PROCEDURE template

A PROCEDURE template is a SAM file that may contain:

– Account number

  Syntax: #OPT:ACCOUNT=<alphanum-name 1..8>

  If present, this value overrides the one of the configuration file.

– Variables

  Variables may be defined as follows:
  – #PAR:*SDF-P procedure variable name*=&&VARIABLE(*variable name*)
  – #PAR:&&VARIABLE(*variable name*)

  It is recommended that you use only one definition form in the same template.

  With the first form you can read in a variable name easily in the SDF-P procedure via the READ-VARIABLE command.

  At run time, &&VARIABLE() is replaced by the current value of the attribute identified by its name.

  The list of the keywords available for the variable name is described in the documentation on the product or the application raising the events. In the context of SPOOL, this information is available in the “SPOOL (BS2000/OSD)” documentation.

  In addition, special variables can be inserted:
  – &&VARIABLE(EVENT-NAME) specifies the event associated with the notification
  – &&VARIABLE(OBJECT-CLASS-NAME)
    specifies the object class name associated with the event
  – &&VARIABLE(EVENT-DATE) specifies the event date in the form YYYY-MM-DD
  – &&VARIABLE(EVENT-TIME) specifies the event time in the form HH:MM:SS

  Template lines that do not respect the rules above are ignored and the warning message SNC0008 is sent to the console.

  Valid variable names that are empty or unknown are replaced by the error string ** no value **.

Example 1

Template text:

```
#OPT:ACCOUNT=1
#PAR:EVENT-NAME=&&VARIABLE(EVENT-NAME)
#PAR:TSN=&&VARIABLE(TSN)
#PAR:USER-NAME=&&VARIABLE(USERID)
```

Resulting parameter file:

```
EVENT-NAME = PRINTJOBACCEPTED
TSN = TSN1
USER-NAME = TSOS
```

Example 2

Template text:

```
#OPT:ACCOUNT=1
#PAR:&&VARIABLE(EVENT-DATE)
#PAR:&&VARIABLE(EVENT-TIME)
```

Resulting parameter file:

```
2004-05-03
16:56:30
```

6.3.6 Example of the PROCEDURE delivery method

This section contains an example of a procedure notification for a print job event.

For example, it is supposed that RSO jobs are waiting to print on the printer PRT0001 and it is not possible to start the printer. The administrator therefore wants to automatically redirect the RSO jobs addressed to printer PRT0001 to printer PRT0002.

The following steps are necessary to do this:

– Define the parameter file layout in a method template.
  To do so, build a SAM file template containing the following information:

```
#OPT:ACCOUNT=1
#PAR:TSN=&&VARIABLE(TSN)
#PAR:PRINTER-NAME=&&VARIABLE(PRINTER-NAME)
```

  Save the file under the name `$TSOS.REDIRECT.SNRTP.TEMPLATE`.

– Write the SDF-P procedure that redirects the print jobs and save it under the name `REDIR.PROC`.

– Define the appropriate method referring to this template:

```
/START-NOTIFICATION-MANAGER
//ADD-NOTIFICATION-RESOURCES
    TYPE=*METHOD(NAME=RDIRPROC,TYPE=BYPROCEDURE,
                 LOCATION=*LIBRARY-ELEMENT(LIBRARY=*STD,ELEMENT=MTHPROC),
                 TEMPLATE=$TSOS.REDIRECT.SNRTP.TEMPLATE,
                 STATE=*ENABLE)
```

– Finally register the subscriptions for RSO jobs:

```
/START-NOTIFICATION-MANAGER
//ADD-NOTIFICATION-RESOURCES
    TYPE=*SUBSCRIPTION(OBJECT-CLASS-NAME=RSOJOB,
                       OBJECT-USER=*ALL,
                       EVENT-NAMES=PRINTJOBACCEPTED,
                       RECIPIENT=*PARAMETERS(ADDRESS=REDIR.PROC,
                                             METHOD-NAME=RDIRPROC))
```

6.4 The FILE delivery method

This notification delivery method is responsible for performing notification by writing the information to a file.

When an event occurs, the Notification Service gathers all the subscriptions interested in this event and addresses the notification information to the FILE delivery method that writes the information to the file associated with the subscription.

The output file where the events are logged is specified in the subscription registration. For security reasons, only the TSOS user ID is able to create a file on a different user ID from its own. If a non-privileged user specifies a user ID other than his/her own, the subscription will be registered but the FILE method will not log the events in the file.

6.4.1 Activating the FILE delivery method

The FILE method is activated when a subscription requiring this method matches a raised event. Such a subscription can be registered in the Notification Resources Manager as follows:

```
/START-NOTIFICATION-MANAGER
//ADD-NOTIFICATION-RESOURCES
    TYPE=*SUBSCRIPTION(OBJECT-CLASS-NAME=...,
                       OBJECT-ID=...,
                       OBJECT-USER=...,
                       OBJECT-ATTRIBUTES=...,
                       EVENT-NAMES=...,
                       USER-DATA=...,
                       RECIPIENT=*PARAMETERS(ADDRESS=file name ,
                                             METHOD-NAME=method name ))
```

where

file name Name of the recipient file. No check is performed upon subscription registration.
This file name must comply with the BS2000 naming conventions (<c-string> cannot be used). The file name may contain catid and userid references. If not, the default catid and userid of the subscription owner are taken.
If the specified name is not a valid file name, an error message is issued and no subscription is registered.

As more than one method can address the same recipient file, synchronization is performed with a job variable. The name of this job variable is the same as the file name (truncated to 46 characters if need be), suffixed with .MTHFILE

If the recipient file is deleted during a session, an error message will be issued at the next relevant event and the method will be disabled.
In this case, you must also erase the JV beforehand to enable the method again. A new recipient file and JV will then be created.

The CHANGE-FILE-NOTIFICATION command allows you to continue with an empty file (see page 82), e.g. for evaluation reasons or if space problems occur.

When a new session is started the recipient file name and JV are initialized again. Do not forget to save the recipient file if necessary.

Note

The job variables used by the method to synchronize access to the notification files will be deleted upon SNRTP subsystem termination by a clean-up batch (job name `MTHFILEC`) started when the method is first called. Synchronization of the clean-up job with the SNRTP subsystem is performed with the $TSOS.SNRTP job variable.

method name Logical delivery method name as defined in the Notification Resources Manager. This method must refer to the FILE element to activate the appropriate method.

6.4.2 FILE notification recipient

The recipient of the notification is a file whose name is specified in a user subscription resource. This user is named “the subscription owner”.

The file layout depends on the template specified in the method definition or is a default layout. See section “FILE template” on page 76 for more details about default layout and templates.

The recipient file (ISAM) has the file attributes
'STD(16), RECFORM=V ,KEYPOS=5, KEYLEN=19'.

It is created when the first relevant event occurs and opened in 'shared update' mode.

6.4.3 Structure of a FILE event notification

This section describes the contents of the file built and extended by the FILE notification delivery method.

The file is composed of two parts: ISAM key and header/notification text.

ISAM key No. 0	Header (when no template is associated)
ISAM key No. 1	Notification text (event 1)
. . .	. . .
ISAM key No. n	Notification text (event n)

Figure 9: Structure of the output file for the FILE notification delivery method

6.4.3.1 ISAM key

The ISAM key has the format: `YYYY-MM-DD HH:MM:SS`

The ISAM key No. 0 (header line) has the format: `YYYY-MM-DD 00:00:00`

Duplicate keys can be present.

6.4.3.2 Header / Notification Text

The header or notification text lines are created on the basis of the template in which the variable names are replaced by their values, taking into account the conditional parts.

Example (after 3 events)

```
EVENT-NAME              OBJECT-CLASS OBJECT-ID        USER-DATA      ... OBJECT-ATTRIBUTES
PRINTJOBACCEPTED        SPOOLJOB     ABCD0000D241ZE20 This is my data... <TSN = ABCD> <USERID = TSOS>
PRINTJOBSTARTED         SPOOLJOB     ABCD0000D241ZE20 This is my data... <TSN = ABCD> <USERID = TSOS>
PRINTJOBCOMPLETED       SPOOLJOB     ABCD0000D241ZE20 This is my data... <TSN = ABCD> <USERID = TSOS>
```

When no template is associated with the method, the first record of the file contains a default header line.

Note

If a template is used, it must specify explicitly where user data and object attributes requested upon subscription have to be located in the file. This is done using special variables. See section “FILE template” on page 76.

6.4.4 Configuration of the FILE delivery method

The administrator can create a configuration file to specify a default header line and a default template.

The file must be present in the $SYSSNS.SYSLIB.SNRTP.METHOD library under the element name MTHFILE.CONFIG of type D.

The following lines can be included in the configuration file:

Option	Description
HEADER=YES | NO	If HEADER=YES is present in the confguration file, the header part will be included in the notification recipient file. If HEADER=NO (or any other value) is present in the configuration file, no header part will be included in the notification recipient file.
FILENAME=*template file name*	File name of the default template file used. This template is used if there is no template associated with the mail method or the template does not exist or is not accessible. This default template can also be a library element. In this case the line FILENAME is replaced by these four lines: LIBRARY=*template library name* ELEMENT=*library element* TYPE=D VERSION=001

If the configuration file is not present or not accessible, a standard header line will be included in the notification recipient file.

6.4.5 FILE template

A FILE template is a SAM file that may contain:

– Header lines

Syntax: `#OPT:HEADER-BEGIN`
`*NONE | <text>`
`#OPT:HEADER-END`

All lines between `#OPT:HEADER-BEGIN` and `#OPT:HEADER-END` are treated as header lines.

`*NONE` specifies that no specific header lines will be written at the beginning of the file. `<text>` specifies that this text will be written at the beginning of the file.

Several header lines may be present. Variable fields or conditional text are not allowed in this area. The header option is interpreted only if `#OPT:HEADER-BEGIN` is the first line of the template file.

– Variable fields

Variable fields are fields that reference an object attribute and that are replaced in the file by the current value of the object attribute. Variable fields must be defined as follows:

`&&VARIABLE(`*variable name*`[,`*length*`])`

where

variable name Name of an object attribute

length Maximum length of data to be displayed. If the value is shorter than the *length*, the data is left-justified and padded with blank characters. If *length* is not specified, the data is written as received.

The list of the keywords available for the variable name is described in the documentation on the product or the application raising the events. In the context of SPOOL, this information is available in the “SPOOL (BS2000/OSD)” documentation.

In addition, the following special variables can be inserted:

– `&&VARIABLE(USER-DATA[,`*length*`])`
Specifies that the user data given upon subscription must be inserted at this place. If this variable is not present in the template, the user data is not included in the recipient file.

– `&&VARIABLE(EVENT-NAME[,`*length*`])`
Specifies the event associated with the notification.

– `&&VARIABLE(OBJECT-CLASS-NAME[,`*length*`])`
Specifies the object class name associated with the event.

  - `&&VARIABLE(OBJECT-ID[,`*length*`])`
    Specifies the object identifier associated with the event.

- Conditional commands that allow conditional text generation. Conditional text generation must be defined as follows:

  `&&IF(`*condition*`) ... <text> ... &&ENDIF()`

  The list of the keywords available for supported conditions is described in the documentation on the product or the application raising the events. In the context of SPOOL, this information is available in the “SPOOL (BS2000/OSD)” documentation.

Lines starting with # and not followed by `OPT:` are ignored. Such lines may be used as comment lines.

Each line of the template file is interpreted and copied to the notification body with the variables being substituted by their values (in string form). Texts included in conditional text will be interpreted and copied only if the condition is satisfied.

In the case of erroneous lines (e.g. syntax error, wrong attributes) an error line is written into the recipient file stating the error condition and suffixed by the wrong line. The interpretation of the template file is stopped in the case of erroneous lines.
For example: `&&VARIABLE()` generates the following line in the recipient file:
`Error in variable substitution : &&VARIABLE()`

If no value is found for variable attributes, blanks are inserted.
For example: `&&VARIABLE(TSN,5)␣&&VARIABLE(LINE-TRUNC)␣&&VARIABLE(USERID)` generates the following line in the recipient file:
`4812 ␣ ␣TSOS`

Example 1

The following template shows an application of the FILE delivery method:

```
#This is a template example.
&&IF(EVENT=PRINTJOBCOMPLETED)
The print job with tsn &&VARIABLE(TSN) has been successfully processed.
&&ENDIF()
&&IF(EVENT=PRINTJOBABORTED)
The print job with tsn &&VARIABLE(TSN) belonging to user
&&VARIABLE(USERID,8) has been cancelled.
&&ENDIF()
```

Resulting file:

```
2003-09-23 16:56:30 The print job with tsn TSN1 has been successfully
processed
2003-09-23 16:56:45 The print job with tsn TSN2 belonging to user
2003-09-23 16:56:45 USER1    has been cancelled.
```

Example 2

The following template includes an explanatory header line:

```
#OPT:HEADER-BEGIN
EVENT                    TSN  USERID
#OPT:HEADER-END
&&VARIABLE(EVENT-NAME,24) &&VARIABLE(TSN,4) &&VARIABLE(USERID,8)
```

Resulting file:

```
2003-09-24 00:00:00 EVENT                    TSN  USERID
2003-09-24 11:19:15 PRINTJOBCOMPLETED        TSN1 TSOS
2003-09-24 11:19:45 PRINTJOBABORTED          TSN2 USER1
```

6.4.6 Example of the FILE delivery method

This section contains an example of a file notification for a print job event.

For example, it is supposed that an administrator wants to log all the events concerning the print jobs of the user ID USER1. He/she wants the log file to have a particular layout.

The following steps are necessary to do this:

– Define the recipient file layout in a method template.
  To do so, build a SAM file template containing the following information:

```
EVENT                    TSN  USERID
&&VARIABLE(EVENT-NAME,24) &&VARIABLE(TSN,4) &&VARIABLE(USERID,8)
```

  Save the file under the name $TSOS.LOGFILE.SNRTP.TEMPLATE.

– Define the appropriate method referring to this template:

```
/START-NOTIFICATION-MANAGER
//ADD-NOTIFICATION-RESOURCES
    TYPE=*METHOD(NAME=LOGFILE,TYPE=TOFILE,
                 LOCATION=*LIBRARY-ELEMENT(LIBRARY=*STD,ELEMENT=MTHFILE),
                 TEMPLATE=$TSOS.LOGFILE.SNRTP.TEMPLATE,
                 STATE=*ENABLE)
```

– Finally, register the subscriptions for SPOOL, RSO and DPRINT jobs:

```
/START-NOTIFICATION-MANAGER
//ADD-NOTIFICATION-RESOURCES
  TYPE=*SUBSCRIPTION(OBJECT-CLASS-NAME=SPOOLJOB,
                     OBJECT-USER=USER1,
                     RECIPIENT=*PARAMETERS(ADDRESS=LOGFILE.SNRTP.TEMPLATE,
                                          METHOD-NAME=LOGFILE))

/START-NOTIFICATION-MANAGER
//ADD-NOTIFICATION-RESOURCES
  TYPE=*SUBSCRIPTION(OBJECT-CLASS-NAME=RSOJOB,
                     OBJECT-USER=USER1,
                     RECIPIENT=*PARAMETERS(ADDRESS=LOGFILE.SNRTP.TEMPLATE,
                                          METHOD-NAME=LOGFILE))

/START-NOTIFICATION-MANAGER
//ADD-NOTIFICATION-RESOURCES
  TYPE=*SUBSCRIPTION(OBJECT-CLASS-NAME=DPRNTJOB,
                     OBJECT-USER=USER1,
                     RECIPIENT=*PARAMETERS(ADDRESS=LOGFILE.SNRTP.TEMPLATE,
                                          METHOD-NAME=LOGFILE))
```

– With each print job created and processed for the user ID `USER1`, the file will be extended. The administrator will, for example, see the following contents:

```
2003-09-24 00:00:00EVENT                    TSN  FILENAME
2003-09-24 12:14:36PRINTJOBACCEPTED         AB01 $USER1.MYFILE.1
2003-09-24 12:14:37PRINTJOBSTARTED          AB01 $USER1.MYFILE.1
2003-09-24 12:15:40PRINTJOBACCEPTED         AB03 $USER1.MYFILE.2
2003-09-24 12:15:45PRINTJOBCOMPLETED        AB01 $USER1.MYFILE.1
2003-09-24 12:15:46PRINTJOBSTARTED          AB03 $USER1.MYFILE.2
2003-09-24 12:16:12PRINTJOBACCEPTED         AB04 $USER1.MYFILE.3
2003-09-24 12:16:30PRINTJOBABORTED          AB04 $USER1.MYFILE.3
2003-09-24 12:17:22PRINTJOBCOMPLETED        AB03 $USER1.MYFILE.2
```

7 SNS commands and statements

This section provides a description of the SNS commands and the Notification Resources Manager statements. The commands and statements have an SDF syntax. You can find a description of the SDF syntax representation in section “SDF syntax representation” on page 159. For a detailed description of SDF see the manual “Introductory Guide to the SDF Dialog Interface“.

7.1 SNS commands

This section describes the SNS commands:

Command	Meaning
CHANGE-FILE-NOTIFICATION	changes the recipient file for a FILE notification delivery method
START-NOTIFICATION-MANAGER	starts the Notification Resources Manager
START-SUBSYSTEM	starts the SNRTP subsystem
STOP-SUBSYSTEM	stops the SNRTP subsystem

CHANGE-FILE-NOTIFICATION - Change recipient file

User group: SNS administrators, nonprivileged users

Privileges: STD-PROCESSING

This command changes the recipient file to be used in a FILE notification delivery method.

This command has no effect for the other SNS notification delivery methods.

Format

CHANGE-FILE-NOTIFICATION
RECIPIENT-FILE-NAME = <filename 1..54>

Operands

RECIPIENT-FILE-NAME = <filename 1..54>
This parameter specifies the name of the notification recipient file to be changed.

Usage notes

1. If no `catid` and/or `userid` are given, the file is searched for under the user ID entering the command.
2. This command closes the recipient file and copies it to a file with the same name (truncated to 34 characters, if need be) suffixed by date and time (`YYYY-MM-DD.HH-MM-SS`).
3. An empty file is opened with the original recipient file name.

START-NOTIFICATION-MANAGER - Starting the Notification Resources Manager

User group: SNS administrators, nonprivileged users

Privileges: STD-PROCESSING

This command starts the Notification Resources Manager.

Format

START-NOTIFICATION-MANAGER
MONJV = ***NONE** / <filename 1..54 without-gen-vers> **,CPU-LIMIT =** ***JOB-REST** / <integer 1..32767>

Operands

MONJV = *NONE / <filename 1..54 without-gen-vers>
Specifies if a job variable is to monitor the job. This operand is only available to users who have the JV software product (see also the “Job Variables” manual).

MONJV = *NONE
No job variable is to monitor the job.

MONJV = <filename 1..54 without-gen-vers>
Name of the job variable to monitor the job.
During the program run, the system sets the job variable to the appropriate values:

$R. the program is running

$T. the program has terminated normally

$A. the program has terminated abnormally

CPU-LIMIT = *JOB-REST / <integer 1..32767>
Maximum CPU time in seconds which the program may use when executing.
If the program execution exceeds the specified time, it is interrupted in interactive mode and message EXC0075 is issued. The user can then request a dump, abort the program or continue the program run. If a STXIT routine has been defined in the program for reaching the CPU limit, this routine is run and the program is terminated. In batch mode the program is terminated.

CPU-LIMIT = *JOB-REST
If the job was started with a time limit, the value defined at system generation is used as the time limit for the program. Otherwise the program runs without a time limit.

CPU-LIMIT = <integer 1..32767>
Maximum CPU time in seconds which the program may use when executing.

Usage notes

1. If the Notification Resources Manager is started while the SNRTP subsystem is not present, any subsequent statement will be rejected and the execution of the Notification Resources Manager will be stopped.
2. If the SNRTP subsystem is stopped while the Notification Resources Manager is loaded, the execution of any statement will be rejected and will stop the Notification Resources Manager.
3. Entering the END statement terminates the Notification Resources Manager.

START-SUBSYSTEM - Starting the SNRTP subsystem

User group: SNS administrators

Privileges: SYSTEM ADMINISTRATION

This command starts the SNRTP subsystem.

Format

START-SUBSYSTEM
SUBSYSTEM-NAME = SNRTP **,VERSION = *STD / '01.0'** **,SUBSYSTEM-PARAMETER = *NONE** **,RESET = *NO** **,SYNCHRONOUS = *NO / *YES** **,VERSION-PARALLELISM = *NONE** **,MONJV = *NONE /** <filename 1..54 without -gen-vers>

Operands

See the “Commands (Volume 1 -5)” manual.

Usage notes

1. The START-SUBSYSTEM command automatically implies the start or the resume of the TU-subsystem SNRKERN.
2. The START-SUBSYSTEM command should be submitted as soon as possible after SYSTEM READY. It should be included in the CMD file as first started subsystem or before the start of the first subsystem that uses it.

STOP-SUBSYSTEM - Stopping the SNRTP subsystem

User group: SNS administrators

Privileges: SYSTEM ADMINISTRATION

This command stops the SNRTP subsystem. A force option provides an immediate deactivation of SNRTP.

Format

STOP-SUBSYSTEM
SUBSYSTEM-NAME = SNRTP **,VERSION = *STD / '01.0'** **,SUBSYSTEM-PARAMETER = *NONE** **,FORCED = *NO / *YES** **,SYNCHRONOUS = *NO / *YES**

Operands

See the “Commands (Volume 1 -5)” manual.

Usage notes

1. In order to ensure a continuity of the Notification Service all along a BS2000 session and in order to prevent from any loss of events and their resulting notifications, the SNRTP subsystem should not be stopped.
2. FORCED=*YES implies the unavailability of SNRTP, as quickly as possible. System dumps may result. It implies also that some events and subsequently resulting notifications may be lost.
3. The stop of the SNRTP subsystem implies that some raised events as well as their resulting notifications may be lost.

7.2 Notification Resources Manager statements

The Notification Resources Manager provides the following statements to manipulate the notifications objects:

Statement	**Meaning**
ADD-NOTIFICATION-RESOURCES	creates a new notification resource
END	terminates the execution of the Notification Resources Manager
MODIFY-NOTIFICATION-RESOURCES	changes the attributes of notification resources
REMOVE-NOTIFICATION-RESOURCES	deletes notification resources
SHOW-NOTIFICATION-RESOURCES	displays the attributes of notification resources

ADD-NOTIFICATION-RESOURCES

User group: SNS administrators

Privileges: SYSTEM ADMINISTRATION
NOTIFICATION ADMINISTRATION
STD PROCESSING for adding a subscription

This statement creates a new notification resource and enters it in the current notification resources file. The type of notification resource to be created is selected via the TYPE operand.

Format

(part 1 of 2)

ADD-NOTIFICATION-RESOURCES

TYPE = ***OBJECT-CLASS**(…) / ***EVENT**(…) / ***METHOD**(…) / ***SUBSCRIPTION**(…)

TYPE = ***OBJECT-CLASS**(…)
NAME = <alphanum-name 1..8>
,**DOMAIN** = <alphanum-name 1..8>
,**STATE** = ***DISABLE** / ***ENABLE**

TYPE = ***EVENT**(…)
NAME = <alphanum-name 1..24>
,**OBJECT-CLASS-NAME** = <alphanum-name 1..8>
,**STATE** = ***DISABLE** / ***ENABLE**
,**TYPE** = ***NORMAL** / ***TERMINAL**

TYPE = ***METHOD**(…)
NAME = <alphanum-name 1..8>
,**TYPE** = <alphanum-name 1..16>
,**LOCATION** = ***LIBRARY-ELEMENT**(…)
***LIBRARY-ELEMENT**(…)
LIBRARY = ***STD** / <filename 1..54 without-vers-gen>
,**ELEMENT** = ***METHOD-NAME** / <alphanum-name 1..8>(…)
<alphanum-name 1..8> (...)
VERSION = *HIGHEST-EXISTING / <composed-name 1..24>
,**TYPE** = ***L**
,**TEMPLATE** = ***NONE** / <filename 1..54 without-vers-gen> / ***LIBRARY-ELEMENT**(…)
***LIBRARY-ELEMENT**(…)
LIBRARY = <filename 1..54 without-vers-gen>
,**ELEMENT** = <composed-name 1..64>(…)
<composed-name 1..64>(…)
VERSION = ***HIGHEST-EXISTING** / <composed-name 1..24
,**TYPE** = <alphanum-name 1..8>
, **STATE** = ***DISABLE** / ***ENABLE**

continued ➠

(part 2 of 2)

```
TYPE = *SUBSCRIPTION(…)
    |  OBJECT-CLASS-NAME = <alphanum-name 1..8>
    |  ,OBJECT-ID = *ALL / <alphanum-name 1..16>
    |  ,OBJECT-USER = *OWN / *ALL / <alphanum-name 1..8>
    |  ,OBJECT-ATTRIBUTES = *NONE / *ALL / list-poss(20): <text 1..64>
    |  ,EVENT-NAMES = *ALL / list-poss(20): <alphanum-name 1..24>
    |  ,USER-DATA = *NONE / <text 1..63 with-low>
    |  ,RECIPIENT = *PARAMETERS(...)
    |     PARAMETERS(...)
    |        |  ADDRESS = <text 1..224 with-low>
    |        |  ,METHOD-NAME = <alphanum-name 1..8>
,COMMENTS = *NONE / <c-string 1..200 with-low>
```

Description of the operands

TYPE = *OBJECT-CLASS(…) / *EVENT(…) / *METHOD(…) / *SUBSCRIPTION(…)
Specifies the type of notification resource that is to be created and whose attributes are to be defined in the corresponding substructure.

TYPE = *OBJECT-CLASS (…)
A notification resource of type object class, on which events can be defined, is to be created.

NAME = <alphanum-name 1..8>
Class of object on which events can be defined, e.g.SPOOLJOB for Spool, RSOJOB for RSO.

DOMAIN = <alphanum-name 1..8>
Name of the concerned domain/product, e.g SPPRINT for the Spool & Print objects. The DOMAIN attribute associated to the object class resource has a particular function. If a module with the same name is found in the system library SYSLIB/SRMLIB/SPMLIB.SNRTP.010.PRIV (hardware-dependent), that means that the product or application the object class is relative to provides a specific product dedicated privilege policy. This policy is automatically loaded when the new object class resource is added.

STATE = *DISABLE / *ENABLE
Specifies whether the object may be used by SNS.

STATE = *DISABLE
The object is only defined, but cannot be used.

STATE = *ENABLE
The object can be used

TYPE = *EVENT (...)
A notification resource of type event is to be created.
When an event resource is added, the relative object class name must exist, otherwise the statement is rejected.

NAME = <alphanum-name 1..24>
Name under which the event object is to be stored, e.g. PRINTJOBCOMPLETED for SPOOLJOB objects

OBJECT-CLASS-NAME = <alphanum-name 1..8>
Associated object class, e.g. SPOOLJOB for the Spool & Print product

STATE = *DISABLE / *ENABLE
Specifies whether the object may be used by SNS.

STATE = *DISABLE
The object is only defined, but cannot be used.

STATE = *ENABLE
The object can be used

TYPE = *NORMAL / *TERMINAL
Specifies the event type

TYPE = *NORMAL
All events that are not terminal

TYPE = *TERMINAL
The event refers to a state change that involves the end of life of an object instance.

Example

In the frame of Spool & Print, the events PRINTJOBCOMPLETED and PRINTJOBABORTED are of type TERMINAL. That means when such events occur, the print job is terminated i.e. the corresponding occurrence of an instance of an object of the class SPOOLJOB ends its life.

TYPE = *METHOD (...)
A notification resource of type notification delivery method is to be created.
When a method resource is added, no specific validation about the library existence, element existence or filename existence is done.

NAME = <alphanum-name 1..8>
Name under which the notification delivery method object is to be stored.

TYPE = <alphanum-name 1..16>
Type of the notification delivery method, see the table on page 26.

LOCATION = *LIBRARY-ELEMENT(...)
Specifies where the notification delivery method is located. The delivery method object module must be saved as an element of a PLAM library.

LIBRARY = *STD / <filename 1..54 without-vers-gen>
Name of the PLAM library in which the notification delivery method is located.

LIBRARY = *STD
The delivery method object module is stored in the PLAM library $SYSSNS.SYSLIB.SNRTP.METHOD.

LIBRARY = <filename 1..54 without-vers-gen>
The delivery method object module is located in the user library with the given filename.

ELEMENT = *METHOD-NAME / <alphanum-name 1..8>(…)
Name of the notification delivery method object module.

ELEMENT = *METHOD-NAME
The name of the object module is the same as the name of the notification delivery method.

ELEMENT = <alphanum-name 1..8>(...)
Name of the delivery method object module, see the table on page 26.

VERSION = *HIGHEST-EXISTING / <composed-name 1..24>
Version of the library element.
If several versions of a method plug-in are defined in a library, we suggest to define the method resource by referencing the precise version of the method element for clarity and performance reasons. Indeed, a plug-in defined with *HIGHEST-EXISTING version will be unloaded and reloaded at each execution.

VERSION = *HIGHEST-EXISTING
The library element with the highest version number is taken.

VERSION = <composed-name 1..24>
The specified version of the library element is taken.

TYPE = *L
Type of the library element. Only library elements of type L are supported for method plug-ins.

TEMPLATE = *NONE / <filename 1..54 without-vers-gen>
/ *LIBRARY-ELEMENT(…)
Location of the template that must be used for the generation of the event notification. Templates used as models for the notification must be SAM files (no check is done), or library elements saved as SAM.

TEMPLATE = *NONE
No template is associated to the notification delivery method

TEMPLATE = <filename 1..54 without-vers-gen>
Name of the SAM file that contains the template.

TEMPLATE = *LIBRARY-ELEMENT(…)
Name of the library element from a PLAM library that contains the template.

LIBRARY = <filename 1..54 without-vers-gen>
Name of the PLAM library.

ELEMENT = <composed-name 1..64>(...)
Name of the library element that contains the template.

VERSION = *HIGHEST-EXISTING / <composed-name 1..24>
version number of the library element

VERSION = *HIGHEST-EXISTING
The highest version of the library element is taken.

VERSION = <composed-name 1..24>
The specified version of the library element is taken.

TYPE = <alphanum-name 1..8>
Type of the library element

STATE = *DISABLE / *ENABLE
Specifies whether the object may be used by SNS

STATE = *DISABLE
The object is only defined, but cannot be used.

STATE = *ENABLE
The object can be used

TYPE = *SUBSCRIPTION (...)
A notification resource of type subscription is to be created. It allows a client to associate subscriptions to a particular object class or instance of an object class.
When a subscription resource is added, referenced object class name, event names and method name must exist, otherwise the statement is rejected.

OBJECT-CLASS-NAME = <alphanum-name 1..8>
Associated object class

OBJECT-ID = *ALL / <alphanum-name 1..16>
Identifier of an object instance

OBJECT-ID = *ALL
All instances of an object class (e.g. all print jobs) are selected depending on the value of the OBJECT-USER operand.

OBJECT-ID = <alphanum-name 1..16>
Identifies a specific object instance, e.g. a specific print job.

OBJECT-USER = *OWN / *ALL / <alphanum-name1..8>
Refers to the user ID the object instance has to belong to.

OBJECT-USER =*OWN
The own user ID is used to identify the owner of the objects

OBJECT-USER =*ALL
The objects of all the users are concerned.

OBJECT-USER = <alphanum-name 1..8>
User ID used to identify the object owner.

OBJECT-ATTRIBUTES = *NONE / *ALL / list-poss(20): <text 1..64>
Objects (e.g. a print jobs) for which notifications are sent may have attributes associated to them. A user may want to have one or more of these associated attributes returned along with a particular notification. In general, these may include any attribute associated to the object emitting the notification. The values of these attributes are provided by the product supporting the notification when it raises the different events. The list of the possible object attribute names should be provided in the documentation of the product supporting the notification.

OBJECT-ATTRIBUTES = *NONE
No attribute is selected

OBJECT-ATTRIBUTES = *ALL
All the attributes associated to the object class are selected.

OBJECT-ATTRIBUTES = list-poss(20): <text 1..64>
Specific attribute names are selected. The names must match exactly with the names defined in the documentation of the product supporting the notification.

EVENT-NAMES = *ALL / list-poss(20): <alphanum-name 1..24>
A list of subscribed events

EVENT-NAMES = *ALL
All the events associated to the object class are selected.

EVENT-NAMES = list-poss(20): <alphanum-name 1..24>
Names of the events that are selected

USER-DATA = *NONE / <text 1..63 with-low>
Contains text that some delivery methods include in each event notification. For details see the relevant sections in the description of the notification delivery methods.

USER-DATA = *NONE
No user data will be included in the notification.

USER-DATA = <text 1..63 with-low>
Text that will be included in the notification.

RECIPIENT = *PARAMETERS(...)
Delivery address for the notifications.

ADDRESS = <text 1..224 with-low>
Contains the recipient address, which depends on the notification delivery method:
- the e-mail address of the notification recipient (MAILTO and OPGMAIL methods)
- the name of the SDF-P procedure to be run (PROCEDURE method)
- the name of the recipient file (FILE method)

METHOD-NAME = <alphanum-name 1..8>
Notification delivery method that is to be used for the notification.

COMMENTS = *NONE / <c-string 1..200 with-low>
Specifies whether any freely definable user comments on the created resource are also to be stored.

COMMENTS = *NONE
No user comments are to be stored.

COMMENTS = <c-string 1..200 with-low>
Specifies the user comment that is to be stored with the created resource.

Statement return codes

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	Successful processing
	64	SNR0008	Internal error
	64	SNR0011	Resource already exists
	64	SNR0012	System interface error
	64	SNR0013	Privileges not sufficient
	64	SNR0016	Referenced resource doesn't exist
	64	CMD0221	System interface error in command

Usage notes

1. The object class, event and method resources can only be added by a privileged user i.e. TSOS or users with the SYSTEM-ADMINSTRATION or NOTIFICATION-ADMINSTRATION privileges. The subscription resources can be added by any users.
   Pay attention that if a user without privilege specifies OBJECT-USER = *ALL or a user ID different from his/her own, he/she will not necessarily receive notifications relative to objects that do not belong to him/her. He/she must respect the privilege policy rules.
2. The resource that is to be created must not yet exist in the notification resources library
   An object class resource and a method resource are identified by their name attributes. The pair (name, object class name) identifies the event object. In the case of subscriptions, the system itself provides an identification attribute.
3. The delivery methods provided with SNS are located in the library $SYSSNS.SYSLIB.SNRTP.METHOD, see the table on page 26.
4. When an object class,event or method resource is disabled, the corresponding subscriptions and subsequently all resulting notifications are ignored until SNS can use it again
5. When a subscription resource is created, an identifier is returned to the user. It is composed of the requestor ID followed by the date and the time. The user must use the ID to modify or remove this subscription.
6. All the resources are saved in the notification resources file NOTIFICATION.PARAMETERS whose location has been determined by IMON at installation.
7. If an event occurs in the system and if several recorded subscriptions require a notification for this event, there will be as many notifications as there are matching subscriptions.

END

User group: SNS administrators, nonprivileged users

Privileges: STD-PROCESSING

This command terminates the Notification Resources Manager.

Format

END

MODIFY-NOTIFICATION-RESOURCES

User group: SNS administrators

Privileges: SYSTEM ADMINISTRATION
NOTIFICATION ADMINISTRATION
STD PROCESSING for adding a subscription

This statement changes the attributes of notification resources, which have been previously saved in the notification resources file. The type of notification resource to be modified is selected via the TYPE operand.

Format

(part 1 of 2)

MODIFY-NOTIFICATION-RESOURCES

TYPE = ***OBJECT-CLASS**(…) / ***EVENT**(…) / ***METHOD**(…) / ***SUBSCRIPTION**(…)

TYPE = ***OBJECT-CLASS**(…)
NAME = <alphanum-name 1..8>
,**NEW-NAME** = ***SAME** / <alphanum-name 1..8>
,**DOMAIN** = ***UNCHANGED** / <alphanum-name 1..8>
,**STATE** = ***UNCHANGED** / ***DISABLE** / ***ENABLE**

TYPE = ***EVENT**(…)
NAME = <alphanum-name 1..24>
,**NEW-NAME** = ***SAME** / <alphanum-name 1..24>
,**OBJECT-CLASS-NAME** = <alphanum-name 1..8>
,**STATE** = ***UNCHANGED** / ***DISABLE** / ***ENABLE**
,**TYPE** = ***UNCHANGED** / ***NORMAL** / ***TERMINAL**

TYPE = ***METHOD**(…)
NAME = <alphanum-name 1..8>
,**NEW-NAME** = ***SAME** / <alphanum-name 1..8>
,**TYPE** = ***UNCHANGED** / <alphanum 1..16>
,**LOCATION** = ***UNCHANGED** / ***LIBRARY-ELEMENT**(…)
***LIBRARY-ELEMENT**(…)
LIBRARY = ***UNCHANGED** / ***STD** / <filename 1..54 without-vers-gen>
,**ELEMENT** = ***UNCHANGED** / ***METHOD-NAME** / <alphanum-name 1..8>(…)
<alphanum-name 1..8 > (...)
VERSION = ***UNCHANGED** / ***HIGHEST-EXISTING /** <composed-name 1..24>
,TYPE = *UNCHANGED / *L
,**TEMPLATE** = ***UNCHANGED** / ***NONE** / <filename 1..54 without-vers-gen> /
***LIBRARY-ELEMENT**(…)
***LIBRARY-ELEMENT**(…)
LIBRARY = ***UNCHANGED** / <filename 1..54 without-vers>
,**ELEMENT** = ***UNCHANGED** / <composed-name 1..64>(…)
<composed-name 1..64>(…)
VERSION = ***UNCHANGED** / ***HIGHEST-EXISTING** / <composed-name 1..24>
,**TYPE** = ***UNCHANGED** / <alphanum-name 1..8>
,**STATE** = ***UNCHANGED** / ***DISABLE** / ***ENABLE**

continued ➠

(part 2 of 2)

```
TYPE = *SUBSCRIPTION(...)
    |  ID = <alphanum-name 20>
    |  ,OBJECT-CLASS-NAME = *UNCHANGED / <alphanum-name 1..8>
    |  ,OBJECT-ID = *UNCHANGED / *ALL/ <alphanum-name 1..16>
    |  ,OBJECT-USER = *UNCHANGED / *OWN / *ALL /<alphanum-name 1..8>
    |  ,OBJECT-ATTRIBUTES = *UNCHANGED / *NONE/*ALL / list-poss(20): <name 1..64>
    |  ,EVENT-NAMES = *UNCHANGED / *ALL / list-poss(20): <name 1..24>
    |  ,USER-DATA = *UNCHANGED / *NONE / <text 1..63 with-low>
    |  ,RECIPIENT = *UNCHANGED / *PARAMETERS(...)
    |     *PARAMETERS(...)
    |        |  ADDRESS = *UNCHANGED / <text 1..224 with-low>
    |        |  ,METHOD-NAME = *UNCHANGED / <alphanum-name 1..8>)
,COMMENTS = *UNCHANGED / *NONE / <c-string 1..200 with-low>
```

Description of the operands

TYPE = *OBJECT-CLASS(...) / *EVENT(...) / *METHOD(...) / *SUBSCRIPTION(...)
Specifies the type of notification resource that is to be changed and whose attributes are to be redefined in the corresponding substructure.

TYPE = *OBJECT-CLASS(...)
A notification resource of type object class, on which events can be defined, is to be modified.

NAME = <alphanum-name 1..8>
Object class name on which events can be defined.

NEW-NAME = *SAME / <alphanum-name 1..8>
Defines whether the name of the notification resource is to be modified.
If a new name is provided for an object class resource, this modification will be rejected if another resource (event or subscription) with this name exists. Such resources must be removed first. See section “REMOVE-NOTIFICATION-RESOURCES” on page 110 for details.
If the new name is the same as the current one, the attribute is ignored and assimilated to *SAME. If the new name refers to an already existing object class the modification will be rejected.

NEW-NAME = *SAME
The name of the notification resource remains unchanged.

NEW-NAME = <alphanum-name 1..8>
New name of the notification resource

DOMAIN = *UNCHANGED / <alphanum-name 1..8>
Name of the concerned domain/product, e.g.SPPRINT for the Spool & Print objects. The DOMAIN attribute associated to the object class resource has a particular function. If a module with the same name is found in the system library SYSLIB/SRMLIB/SPMLIB.SNRTP.010.PRIV (hardware-dependent), that means that the product or application the object class is relative to provides a specific product dedicated privilege policy. This policy is automatically loaded when the new object class resource is added.
Modifying the DOMAIN attributes involves the unload of the previous dedicated privilege policy and the load of the new one if any.

DOMAIN = *UNCHANGED
The last value set for DOMAIN remains valid.

DOMAIN = <alphanum-name 1..8>
Name of the new associated domain

STATE = *UNCHANGED / *DISABLE / *ENABLE
Specifies whether the object may be used by SNS.

STATE = *UNCHANGED
The last value set remains valid.

STATE = *DISABLE
The object is only defined, but cannot be used.

STATE = *ENABLE
The object can be used

TYPE = *EVENT (...)
An event resource is to be modified.

NAME = <alphanum-name 1..24>
Name under which the event resource is stored.

NEW-NAME = *SAME / <alphanum-name 1..24>
Defines whether the name of the notification resource is to be modified.

NEW-NAME = *SAME
The name of the notification resource remains unchanged.

NEW-NAME = <alphanum-name 1..24>
New name of the notification resource.
If a new name is provided for an event resource, the modification will be rejected if another existing resource (subscription) explicitly refers to the old name. Such resources must be removed first. See section “REMOVE-NOTIFICATION-RESOURCES” on page 110 for details. If the new name is the same as the current one, the attribute is ignored and assimilated to *SAME.
If the new name refers to an already existing event the modification will be rejected.

OBJECT-CLASS-NAME = <alphanum-name 1..8>
Refers to the object class on which events can be defined
This operand must always be specified since the duplet NAME/OBJECT-CLASS-NAME unequivocally identifies an event object.

STATE = *UNCHANGED / *DISABLE / *ENABLE
Specifies whether the object may be used by SNS.

STATE = *UNCHANGED
The last value set remains valid.

STATE = *DISABLE
The object is only defined, but cannot be used.

STATE = *ENABLE
The object can be used

TYPE = *UNCHANGED / *NORMAL / *TERMINAL
Specifies the event type

TYPE = *UNCHANGED
The last value set remains valid.

TYPE = *NORMAL
All events that are not terminal

TYPE = *TERMINAL
The event refers to a state change that involves the end of life of an object instance.

Example

In the frame of Spool & Print, the events PRINTJOBCOMPLETED and PRINTJOBABORTED are of type TERMINAL. That means when such events occur, the print job is terminated i.e. the corresponding occurrence of an instance of an object of the class SPOOLJOB ends its life.

TYPE = *METHOD (…)
A notification resource of type notification delivery method is to be modified.

NAME = <alphanum-name 1..8>
Name under which the notification delivery method resource is stored.

NEW-NAME = *SAME / <alphanum-name 1..8>
Defines whether the name of the notification resource is to be modified.

NEW-NAME = *SAME
The name of the notification resource remains unchanged.

NEW-NAME = <alphanum-name 1..8>
New name of the notification resource
If a new name is provided for a method resource, the modification will be rejected if another existing resource (subscription) explicitly refers to the old name. Such resources must be removed first. See section “REMOVE-NOTIFICATION-RESOURCES” on page 110 for details.
If the new name is the same as the current one, the attribute is ignored and assimilated to *SAME.
If the new name refers to an already existing method the modification will be rejected.

TYPE = *UNCHANGED / <alphanum 1..16>
Defines whether the type of the notification delivery method is to be modified.

TYPE = *UNCHANGED
The type is not to be modified. The last value set remains valid.

TYPE = <alphanum 1..16>
New associated type, see the table on page 26.

LOCATION = *UNCHANGED / *LIBRARY-ELEMENT(...)
Specifies whether the location of the notification delivery method object module is to be modified.

LOCATION = *UNCHANGED
The location is not affected by the modification The last value set remains valid.

LOCATION = *LIBRARY-ELEMENT(...)
The specified element from a PLAM library must be used.

LIBRARY = *UNCHANGED / *STD / <filename 1..54 without-vers-gen>
Name of the PLAM library in which the notification delivery method is located.

LIBRARY = *UNCHANGED
The library name is not affected by the modification The last value set remains valid.

LIBRARY = *STD
The delivery method object module is stored in the PLAM library $SYSSNS.SYSLIB.SNRTP.METHOD.

LIBRARY = <filename 1..54 without-vers-gen>
The delivery method object module is located in the user library with the given filename.
No specific validation of the library existence, element existence or filename existence is done.

ELEMENT = *UNCHANGED / *METHOD-NAME / <alphanum-name 1..8>(...)
Name of the notification delivery method object module (library element of type L only).

ELEMENT = *UNCHANGED
The object module name is modified. The last value set remains valid.

ELEMENT = *METHOD-NAME
The name of the object module is the same as the name of the notification delivery method.

ELEMENT = <alphanum-name 1..8>(...)
New name of the delivery method object module, see the table on page 26.

VERSION = *UNCHANGED / *HIGHEST-EXISTING / <composed-name 1..24>
Version of the library element.

VERSION = *UNCHANGED
The element version is not modified. The last value remains valid

VERSION = *HIGHEST-EXISTING
The library element with the highest version number is taken.

VERSION = <composed-name 1..24>
The specified version of the library element is taken.

TYPE = *UNCHANGED / *L
TYPE specifies the type of the library element. *UNCHANGED specifies that the type is not modified and that the last value remains valid. Only library elements of type L are supported for method plug-ins.

TEMPLATE = *UNCHANGED / *NONE / <filename 1..54 without-vers-gen> / *LIBRARY-ELEMENT(...)
Specifies whether the template usage is to be modified.
Templates used as models for the notification must be SAM files (no check is done), or library elements saved as SAM.

TEMPLATE = *UNCHANGED
The template usage is not modified. The last value set remains valid.

TEMPLATE = *NONE
No template is associated to the notification delivery method.

TEMPLATE = <filename 1..54 without-vers-gen>
Name of the SAM file that contains the template.

TEMPLATE = *LIBRARY-ELEMENT(…)
Name of the library element from a PLAM library that contains the template.

LIBRARY = *UNCHANGED / <filename 1..54 without-vers>
Name of the PLAM library

LIBRARY = *UNCHANGED
The library name is not modified. The last value set remains valid.

LIBRARY = <filename 1..54 without-vers>
New library name

ELEMENT = *UNCHANGED / <composed-name 1..64>(…)
Name of the library element that contains the template.

ELEMENT = *UNCHANGED
The name of the library element is not modified. The last value set remains valid.

ELEMENT =<composed-name 1..64>(…)
New name of the library element.

VERSION = *UNCHANGED / *HIGHEST-EXISTING / <composed-name 1..24>
Version of the library element

VERSION = *UNCHANGED
The version is not modified. The last value set remains valid.

VERSION =*HIGHEST-EXISTING
The library element with the highest version number is taken.

VERSION =<composed-name 1..24>
The specified version of the library element is taken.

TYPE = *UNCHANGED / <alphanum-name 1..8>
Type of the library element

TYPE = *UNCHANGED
The type of the library element is not modified. The last value set remains valid.

TYPE = <alphanum-name 1..8>
New element type

TYPE = *SUBSCRIPTION (…)
A notification resource of type subscription, which contains objects of type job subscription, is to be modified.
When a subscription resource is modified, the eventual new referenced object class name, event names and method name must exist, otherwise the modification is rejected.

ID = <alphanum-name 20>
Subscription object that is to be modified. <alphanum-name 20> is the ID of the individual notification resource that is to be modified.

OBJECT-CLASS-NAME = *UNCHANGED / <alphanum-name 1..8>
Associated object.

OBJECT-CLASS-NAME = *UNCHANGED
The object is not modified. The last value set remains valid.

OBJECT-CLASS-NAME = <alphanum-name 1..8>
Name of the new associated object.

OBJECT-ID = *UNCHANGED / *ALL / <alphanum-name 1..16>
Identifier of the object. In case of a subscription for a print job, it is the TSN of the print job.

OBJECT-ID = *UNCHANGED
The object ID is not modified. The last value set remains valid.

OBJECT-ID = *ALL
All the requestor's print jobs are selected.

OBJECT-ID = <alphanum-name 1..16>
Identifies a specific object instance.

OBJECT-USER = *UNCHANGED / *OWN / *ALL / <alphanum-name 1..8>
Refers to the user ID the object instance has to belong to.

OBJECT-USER = *UNCHANGED
The requestor is not affected by the modification. The last value set remains valid.

OBJECT-USER = *OWN
The own user ID is used to identify the requestor.

OBJECT-USER = *ALL
The objects of all the users are concerned. This value is only permitted to the system administrator

OBJECT-USER = <alphanum-name 1..8>
User ID used to identify the object owner

OBJECT-ATTRIBUTES = *UNCHANGED / *NONE /*ALL /
list-poss(20): <name 1..64>
Specifies whether the subscribed attributes are to be modified.

OBJECT-ATTRIBUTES = *UNCHANGED
The subscribed attributes are not modified. The last value set remains valid.

OBJECT-ATTRIBUTES = *NONE
No attribute is selected.

OBJECT-ATTRIBUTES = *ALL
All the attributes associated to the object class are selected.

OBJECT-ATTRIBUTES = list-poss(20): <name 1..64>
Specific attribute names are selected.

EVENT-NAMES = *UNCHANGED / *ALL / list-poss(20): <name 1..24>
Specifies whether the subscribed events are to be modified

EVENT-NAMES = *UNCHANGED
The subscribed events are not modified. The last value set remains valid.

EVENT-NAMES = *ALL
All the events associated to the object classare selected.

EVENT-NAMES = list-poss(20): <name 1..24>
Names of the events that are selected

USER-DATA = *UNCHANGED / *NONE / <text 1..63 with-low>
Specifies whether text that some delivery methods include in each event notification are to be modified. For details see the relevant sections in the description of the notification delivery methods.

USER-DATA = *UNCHANGED
The user data is not modified. The last value set remains valid.

USER-DATA = *NONE
No user data are included in the notification.

USER-DATA = <text 1..63 with-low>
Text that will be included in the notification.

RECIPIENT = *UNCHANGED / *PARAMETERS(...)
Specifies whether the delivery address and/or the notification delivery method is to be modified

RECIPIENT = *UNCHANGED
No modification is requested. The last value set remains valid.

RECIPIENT = *PARAMETERS(...)
The delivery address and/or the notification delivery method is to be modified.

ADDRESS = *UNCHANGED / <text 1..224 with-low>
Specifies whether the delivery address is to be modified.

ADDRESS = *UNCHANGED
The address is not modified. The last value set remains valid.

ADDRESS = <text 1..224 with-low>
The new recipient address, which depends on the notification delivery method:
– the e-mail address of the notification recipient (MAILTO and OPGMAIL methods)
– the name of the SDF-P procedure to be run (PROCEDURE method)
– the name of the recipient file (FILE method)

METHOD-NAME = *UNCHANGED / <alphanum-name 1..8>
Specifies whether the notification delivery method is to be modified.

METHOD-NAME = *UNCHANGED
The notification delivery method is not modified. The last value set remains valid.

METHOD-NAME = <alphanum-name 1..8>
Name of the new notification delivery method that is to be used for the notification.

COMMENTS = *UNCHANGED / *NONE / <c-string 1..200 with-low>
Specifies whether any freely definable user comments on the created resource are to be modified.

COMMENTS = *UNCHANGED
The user comments are not modified. The last value set remains valid

COMMENTS = *NONE
No user comments are stored.

COMMENTS = <c-string 1..200 with-low>
Specifies the user comment that is to be modified.

Statement return codes

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	Successful processing
	64	SNR0008	Internal error
	64	SNR0010	No resource(s) found
	64	SNR0011	Resource already exists
	64	SNR0012	System interface error
	64	SNR0013	Privileges not sufficient
	64	SNR0014	Resource referenced by others objects
	64	SNR0016	Referenced resource doesn't exist
	64	CMD0221	System interface error in command

Usage notes

1. The object class, event and method resources can only be modified by a privileged user i.e. TSOS or users with the SYSTEM-ADMINSTRATION or NOTIFICATION-ADMINSTRATION privileges. The subscription resources modification is available to any user. However, a nonprivileged user is only allowed to modify his/her own subscriptions while a privileged user is allowed to modify any subscriptions.
   Pay attention that if a user without privilege specifies OBJECT-USER = *ALL or a user ID that is different from his/her own, he/she will not necessarily receive notifications relative to objects that do not belong to him/her. He/she must respect the privilege policy rules.
2. When a subscription resource is modified, the eventual new referenced object class name, event names and method name must exist, otherwise the modification is rejected.
3. The delivery methods provided with SNS are located in the library $SYSSNS.SYSLIB.SNRTP.METHOD, see the table on page 26.
4. When an object class, event or method resource is disabled, the corresponding subscriptions and subsequently all resulting notifications are ignored until SNS can use it again.

REMOVE-NOTIFICATION-RESOURCES

User group: SNS administrators

Privileges: SYSTEM ADMINISTRATION
NOTIFICATION ADMINISTRATION
STD PROCESSING for subscriptions

This statement deletes individual notification resources or a set of notification resources matching some selection criteria from the notification resources library.

Format

REMOVE-NOTIFICATION-RESOURCES

TYPE = ***OBJECT-CLASS**(…) / ***EVENT**(…) / ***METHOD**(…) / ***SUBSCRIPTION**(…)

TYPE = ***OBJECT-CLASS**(…)
NAME = ***ALL** / <alphanum-name 1..8 with-wild(24)>

TYPE = ***EVENT**(…)
NAME = ***ALL** / <alphanum-name 1..24 with-wild(48)>
,**OBJECT-CLASS-NAME** = **_*ALL_** / <alphanum-name 1..8 with-wild(24)>

TYPE = ***METHOD**(…)
NAME = ***ALL** / <alphanum-name 1..8 with-wild(24)>

TYPE = ***SUBSCRIPTION**(…)
ID = ***ALL** / <alphanum-name 20 with-wild(48)>
,**OBJECT-CLASS-NAME** = **_*ALL_** / <alphanum-name 1..8 with-wild(24)>
,**OWNER** = **_*OWN_** / ***ALL** / <alphanum-name 1..8>

,**ACCESS-DATE** = **_*ANY_** / ***TODAY** / ***INTERVAL**(…) / <date 8..10>

***INTERVAL** (…)
FROM = **_1950-01-01_** / <date 8..10>
TO = **_*TODAY_** / <date 8..10>

Description of the operands

TYPE = *OBJECT-CLASS(…) / *EVENT(…) / *METHOD(…) / *SUBSCRIPTION(…)
Type of notification resources to be deleted from the notification resource library.

TYPE = *OBJECT-CLASS(…)
Object class resources are to be deleted.
Deletion of an object class is only allowed when no other resources (events or subscriptions) refer to it.

NAME = *ALL / <alphanum-name 1..8 with-wild(24)>
Specifies which object class resources are to be deleted

NAME = *ALL
All object classes are to be deleted

NAME = <alphanum-name 1..8 with-wild(24)>
Name of the individual object class which is to be deleted

TYPE = *EVENT(…)
Event objects are to be deleted. Deletion of an event is only allowed when no subscription resources refer to it.

NAME = *ALL / <alphanum-name 1..24 with-wild(48)>
Specifies which event objects are to be deleted

NAME = *ALL
All events objects are to be deleted

NAME =<alphanum-name 1..24 with-wild(48)>
All matching event objects have to be deleted.

OBJECT-CLASS-NAME = *ALL / <alphanum-name 1..8 with-wild(24)>
OBJECT-CLASS-NAME refers to the associated object

NAME	OBJECT-CLASS-NAME	Meaning
*ALL	Object name	All event resources associated to the given object class are to be deleted
Event name	*ALL	All event resources with the specified name are to be deleted
Event name	Object name	The selected event resource is to be deleted

TYPE = *METHOD(…)
Notification delivery method resources are to be deleted. Deletion of a method is only allowed when no subscription resources refer to it.

NAME = *ALL / <alphanum-name 1..8 with-wild(24)>
Specifies which notification delivery method resources are to be deleted

NAME = *ALL
All notification delivery method objects are to be deleted

NAME = <alphanum-name 1..8 with-wild(24)>
Name of the individual notification delivery method object which is to be deleted

TYPE = *SUBSCRIPTION(…)
Subscription resources are to be deleted.

ID = *ALL / <alphanum-name 20 with-wild(48)>
Identifies the subscriptions which are to be deleted.

ID = *ALL
All subscriptions are to be deleted.

ID = <alphanum-name 20 with-wild(48)>
Id of the individual subscription that is to be deleted.

OBJECT-CLASS-NAME = *ALL / <alphanum-name 1..8 with-wild(24)>
OBJECT-CLASS-NAME specifies that the subscriptions belonging to this object class are to be deleted.

OBJECT-CLASS-NAME = *ALL
The subscriptions belonging to any object class are to be deleted.

OBJECT-CLASS-NAME = <alphanum-name 1..8 with-wild(24)>
The subscriptions belonging to the specified object class are to be deleted.

OWNER = *OWN / *ALL / <alphanum-name 1..8>
Specifies that the subscriptions belonging to this user must be deleted.

OWNER = *OWN
Only the subscriptions belonging to the calling user are to be deleted.

OWNER =*ALL
All subscriptions whatever their owner are to be deleted.

OWNER = <alphanum-name 1..8>
The subscriptions belonging to that particular user have to be deleted. This value is reserved for privileged users. The statement is rejected for unprivileged users.

ACCESS-DATE = *ANY / *TODAY / *INTERVAL(…) / <date 8..10>
The date of the last access can act as an additional criterion for selection of the objects to be deleted. Here you can specify a particular date or a range of dates. Each write access (ADD, MODIFY) to an object by the TU program and each access to a resource in the frame of a notification processing is logged with a timestamp. The access date thus corresponds to the last access to this object.

ACCESS-DATE = *ANY
The access date is not to serve as a criterion for the selection of the objects to be deleted.

ACCESS-DATE = *TODAY
In conjunction with the above-named criteria, the last notification resources to be accessed on the same day are to be deleted.

ACCESS-DATE = *INTERVAL(…)
A range of dates is specified.

FROM = 1950-01-01 / <date 8..10>
Lower limit of the date range. The default value is intended to ensure that no notification resources older than 1950 are included.

TO = *TODAY / <date 8..10>
Upper limit of the date range. The default value is always the current date.

ACCESS-DATE = <date 8..10>
In conjunction with the above-named criteria, the notification resources last accessed on the date specified in the format YY-MM-DD or YYYY-MM-DD are to be deleted.

Statement return codes

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	Successful processing
	64	SNR0008	Internal error
	64	SNR0010	No resource(s) found
	64	SNR0012	System interface error
	64	SNR0013	Privileges not sufficient
	64	SNR0014	Resource referenced by others objects
	64	SNR0016	Referenced resource doesn't exist
	64	CMD0221	System interface error in command

Usage note

The object class, event and method resources can only be deleted by a privileged user i.e. TSOS or users with the SYSTEM-ADMINSTRATION or NOTIFICATION-ADMINSTRATION privileges. Subscription resources can be deleted by any user. However, a non privileged user is only allowed to delete his/her own subscriptions while a privileged user is allowed to delete any subscriptions.

SHOW-NOTIFICATION-RESOURCES

User group: SNS administrators, nonprivileged users

Privileges: SYSTEM ADMINISTRATION
NOTIFICATION ADMINISTRATION
STD-PROCESSING

This statement displays the attributes of individual notification resources or all notification resources in the notification resources library. The display is provided either in summarized form or in full information form.

Format

SHOW-NOTIFICATION-RESOURCES

TYPE = **<u>*ALL</u>** / ***OBJECT-CLASS**(…) / ***EVENT**(…) / ***METHOD**(…) / ***SUBSCRIPTION**(…)

TYPE = ***OBJECT-CLASS**(…)
NAME = **<u>*ALL</u>** / <alphanum-name 1..8 with-wild(24)>

TYPE = ***EVENT**(…)
NAME = **<u>*ALL</u>** / <alphanum-name 1..24 with-wild(48)>
,**OBJECT-CLASS-NAME** = **<u>*ALL</u>** / <alphanum-name 1..8 with-wild(24)>

TYPE = ***METHOD**(…)
NAME = **<u>*ALL</u>** / <alphanum-name 1..8 with-wild(24)>

TYPE = ***SUBSCRIPTION**(…)
ID = **<u>*ALL</u>** / <alphanum-name 20 with-wild(48)>
,**OWNER** = **<u>*OWN</u>** / ***ALL** / <alphanum-name 1..8>
,**OBJECT-CLASS-NAME** = **<u>*ALL</u>** / <alphanum-name 1..8>
,**METHOD-NAME** = **<u>*ALL</u>** / <alphanum-name 1..8>
,**EVENT-NAME** = **<u>*ANY</u>** / ***ALL** / <alphanum-name 1..24>

,**INFORMATION** = **<u>*SUMMARY</u>** / ***ALL**

Description of the operands

TYPE = *ALL / *OBJECT-CLASS(…) / *EVENT(…) / *METHOD(…) / *SUBSCRIPTION(…)
Type of notification resource whose attributes are to be displayed.

TYPE = *ALL
The entire content of the notification resources library is to be displayed except for the subscriptions for which only the ones belonging to the calling user are displayed.

TYPE =*OBJECT-CLASS(…)
Information on object classes is to be displayed.

NAME = *ALL / <alphanum-name 1..8 with-wild(24)>
Object classes about which information is to be displayed.

NAME = *ALL
Information on all object classes is to be displayed.

NAME = <alphanum-name 1..8 with-wild(24)>
Name of the individual object class about which information is to be displayed.

TYPE = *EVENT(…)
Information on event resources is to be displayed.
NAME refers to the event resources about which information is requested.
OBJECT-CLASS-NAME refers to the associated object class.

NAME	OBJECT-CLASS-NAME	Meaning
*ALL	*ALL	Information on all event resources is to be displayed
*ALL	object class name	Information on all event resources associated to the given object class is to be displayed
Event name	*ALL	Information on the event resource with the specified name is to be displayed whatever the object class name is
Event name	object class name	Information on the fully identified event resource is to be displayed

TYPE = *METHOD(…)
Information on notification delivery method resources is to be displayed

NAME = *ALL / <alphanum-name 1..8 with-wild(24)>
Notification delivery method resources about which information is to be displayed.

NAME = *ALL
Information on all notification delivery method resources is to be displayed.

NAME = <alphanum-name 1..8 with-wild(24)>
Name of the individual notification resource about which information is to be displayed.

TYPE =*SUBSCRIPTION(…)
Information on subscription resources is to be displayed.

ID = *ALL / <alphanum-name 20 with-wild(48)>
ID identifies the subscriptions of which the attributes are to be displayed.

ID = *ALL
Information on all subscriptions is to be displayed.

ID = <alphanum-name 20 with-wild(48)>
Id of the individual subscription resource of which information is to be displayed.

OWNER = *OWN / *ALL / <alphanum-name 1..8>
Specifies the user whose subscriptions are to be displayed.

OWNER = *OWN
Only the subscriptions belonging to the calling user are to be displayed.

OWNER = *ALL
All subscriptions whatever their owner are to be displayed. This value is reserved for the system administrator, TSOS, or NOTIFICATION-ADMINISTRATION privileged user.

OWNER = <alphanum-name 1..8>
The subscriptions belonging to the specified user are to be displayed. This value is reserved to the system administrator, TSOS or NOTIFICATION-ADMINISTRATION privileged user.

OBJECT-CLASS-NAME = *ALL / <alphanum-name 1..8>
The subscriptions referring to this specific object class name are to be displayed.

OBJECT-CLASS-NAME = *ALL
All subscriptions whatever their relative object class name are to be displayed.

OBJECT-CLASS-NAME = <alphanum-name 1..8>
All subscriptions relative to this specific object class name are to be displayed.

METHOD-NAME = *ALL / <alphanum-name 1..8>
The subscriptions referring to this specific method name are to be displayed.

METHOD-NAME = *ALL
All subscriptions whatever their relative method name are to be displayed.

METHOD-NAME = <alphanum-name 1..8>
All the subscriptions relative to this specific method name are to be displayed.

EVENT-NAME = *ANY / *ALL / <alphanum-name 1..24>
The subscriptions referring to this specific event name are to be displayed.

EVENT-NAME = *ANY
All subscriptions whatever their relative event names are to be displayed.

EVENT-NAME =*ALL
All subscriptions referring explicitly to all events have to be displayed.

EVENT-NAME = <alphanum-name 1..24>
All subscriptions referring explicitly to this specific event name have to be displayed.

INFORMATION = *SUMMARY / *ALL
Specifies the scope of the information to be displayed.

INFORMATION = *SUMMARY
A summary is provided. Special attributes of the notification resource are not displayed.

INFORMATION = *ALL
All available information on the desired notification resource are displayed;

Statement return codes

(SC2)	SC1	Maincode	Meaning/Guaranteed messages
	0	CMD0001	Successful processing
	64	SNR0008	Internal error
	64	SNR0010	No resource(s) found
	64	SNR0012	System interface error
	64	SNR0013	Privileges not sufficient
	64	SNR0016	Referenced resource doesn't exist
	64	CMD0221	System interface error in command

Usage note

The display of notification resources of types object class, event and notification delivery method is accessible to all user groups.
The display of subscription resources is also accessible to everyone but with different visibility according to the privileges of the requestor.
A privileged user i.e.TSOS, SYSTEM-ADMINISTRATION privileged user or NOTIFICATION-ADMINSTRATION privileged user are allowed to view all the subscriptions of every user. All other users are only allowed to view their own subscriptions no matter what the OWNER parameter value is. If they specify OWNER =*ALL, this is assimilated to OWNER =*OWN and if they specify a specific user ID, the statement is rejected.

Output

Output for OBJECT-CLASS

```
INFO=SUMMARY

OBJECT-CLASS(ES)
------------------------------------------------------------------------------
NAME         DOMAIN     STATE
xxxxxxxx     xxxxxxxx   xxxxxxxx (1)

= *DISABLE or *ENABLE


INFO=ALL

TYPE               :  *OBJECT-CLASS
NAME               :  xxxxxxxx
DOMAIN             :  xxxxxxxx
STATE              :  xxxxxxxx                                              (1)
OWNER              :  xxxxxxxx
ACCESS-DATE        :  xxxx-xx-xx
COMMENTS           :  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (2)
                      xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                      xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                      xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

(1) = *DISABLE or *ENABLE
(2) = *NONE or <text 1..200>

Output for EVENT

```
INFO=SUMMARY

EVENT(S)
-------------------------------------------------------------------------------
NAME                        OBJECT-NAME  STATE     TYPE
xxxxxxxxxxxxxxxxxxxxxxxx    xxxxxxxx     xxxxxxxx  xxxxxxxxx (1)(2)

State = *DISABLE or *ENABLE
Type = *NORMAL or *TERMINAL

INFO=ALL

TYPE                :  *EVENT
NAME                :  xxxxxxxxxxxxxxxxxxxxxxx
OBJECT-NAME         :  xxxxxxxx
STATE               :  xxxxxxxx                                              (1)
TYPE                :  xxxxxxxxx                                             (2)
OWNER               :  xxxxxxxx
ACCESS-DATE         :  xxxx-xx-xx
COMMENTS            :  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (3)
                       xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                       xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                       xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

(1) = *DISABLE or *ENABLE
(2) = *NORMAL or *TERMINAL
(3) = *NONE or <text 1..200>

Output for METHOD

```
INFO=SUMMARY

METHOD(S)
------------------------------------------------------------------------------
NAME      TYPE              STATE
xxxxxxxx  xxxxxxxxxxxxxxxx  xxxxxxxx (1)

(1) = *ENABLE / *DISABLE

INFO=ALL

TYPE               :  *METHOD
NAME               :  xxxxxxxx
TYPE               :  xxxxxxxxxxxxxxxx
LOCATION           :  *LIBRARY-ELEMENT
     LIBRARY       :  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (1)
     ELEMENT       :  xxxxxxxx
       VERSION     :  xxxxxxxxxxxxxxxxxxxxxxx                                  (2)
     TYPE          :  = *L
TEMPLATE           :  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (3)
     LIBRARY       :  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (4)
     ELEMENT       :  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                      xxxxxxxxxx                                               (5)
       VERSION     :  xxxxxxxxxxxxxxxxxxxxxxx                                  (6)
     TYPE          :  xxxxxxxx
STATE              :  xxxxxxxx                                                 (7)
OWNER              :  xxxxxxxx
ACCESS-DATE        :  xxxx-xx-xx
COMMENTS           :  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx     (8)
                      xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                      xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                      xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

(1) = *STD or <filename 1..54>
(2) = *HIGHEST-EXISTING or <composed-name 1..24>
(3) = *NONE or *LIBRARY-ELEMENT or <filename 1..54>
(4) = empty or <filename 1..54>
(5) = empty or <element name 1..64>
(6) = empty or *HIGHEST-EXISTING or <version 1..24>
(7) = *DISABLE/*ENABLE
(8) = *NONE or <text 1..200>

Output for SUBSCRIPTION

```
INFO=SUMMARY

SUBSCRIPTION(S)
------------------------------------------------------------------------------
ID                    OBJECT-NAME  OBJECT-ID         METHOD-NAME   OWNER
xxxxxxxxxxxxxxxxxxxx  xxxxxxxx     xxxxxxxxxxxxxxxx  xxxxxxxx      xxxxxx

INFO=ALL

TYPE                 :  *SUBSCRIPTION
ID                   :  xxxxxxxxxxxxxxxxxxxx
OBJECT-NAME          :  xxxxxxxx
OBJECT-ID            :  xxxxxxxxxxxxxxxx
OBJECT-USER          :  xxxxxxxx
OBJECT-ATTRIBUTES    :  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                        xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (1)
EVENT-NAMES          :  xxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxx (2)
                        xxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxx
                        xxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxx
                        xxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxx
                        xxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxx
                        xxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxx
                        xxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxx
                        xxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxx
                        xxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxx
                        xxxxxxxxxxxxxxxxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxx
USER-DATA            :  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (3)
RECIPIENT-ADDRESS    :  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
                    xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                    xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                    xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
METHOD-NAME       :  xxxxxxxx
OWNER             :  xxxxxxxx
ACCESS-DATE       :  xxxx-xx-xx
COMMENTS          :  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (4)
                     xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                     xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                     xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

(1) = *NONE or *ALL or <list of 20 alphanum 1..64>
(2) = *ALL or <list of 20 alphanum 1..24>
(3) = *NONE or <text 1..63>
(4) = *NONE or <text 1..200>

8 Notification Service APIs

SNS provides the following APIs:

API	Meaning
SNPEVS	describes the parameter list for creating a subscription
SNPREQT	allows user and system access to SNS
SNPREV	describes the parameter list for raising an event

8.1 SNPEVS - Create subscription interface

User group: Nonprivileged users
Programming languages: Assembler, C, CPP
Macro type: S / (C, D, I, L, M)

This interface describes the parameter list allowing the creation of a temporary subscription from any BS2000 product or application. The call to SNS must be done by the SNPREQT interface.

Format

Operation	Operands
SNPEVS	,MF = C / D / I / L / M
	,PREFIX = S / SNPEVS / <name>
	,MACID = <name 1..3>
	,PARAM = <name 1..27>
	,OBJECT_USER = *OWN / *ALL / <var: char 1..8> / <c-string 1..8>
	,OBJECT_NAME = <var: char 1..8> / <c-string 1..8>
	,OBJECT_ID = <var: char 1..16> / <c-string 1..16>
	,DELIVERY_ADDR = <var: char 1..224> / <c-string 1..224>
	,DELIVERY_METH = <var: char 1..8> / <c-string 1..8>
	,EVENTS_TYP = *ALL / *LIST
	,EVENTS = array(20) elem: <c-string 1..24> / <var: char 1..24>
	,ATTRIBUTES_TYP = *NONE / *ALL / *LIST
	,ATTRIBUTES = array(20) elem: <c-string 1..64> / <var: char 1..64>
	,USER_DATA = <var: char 1..63>/ <c-string 1..63> / *NONE
	,COMMENTS = <var: char 1..200> / <c-string 1..200> / *NONE
	,SUB_MODE = *ACCEPT_NOTIFICATION / *REJECT_NOTIFICATION

Description of the operands

MF = C / D / I / L / M
Type of the macro call. You will find more information in the “Executive Macros“ manual.
Possible values for the different programming languages:

Assembler:	C, D, L, M
C:	--
CPP:	D, I, M

Note

You can only specify pointer variables (“var:pointer” operand value) with MF=M. In addition, a list must be generated with MF=D or MF=C.

PREFIX = S / SNPEVS / <name>
Specifies the first character of field names and equates.
Default values for the different programming languages:

Assembler:	S
C:	--
CPP:	SNPEVS

MACID = <name 1..3>
Specifies the second to fourth characters (inclusive) of the field names and equates.
Default values for the different programming languages:

Assembler:	EVS
C:	--
CPP:	--

PARAM = <name 1..27>
Specifies the address of the operand list (permitted only in the case of MF formats 2 and 3). You will find more information in the “Executive Macros” manual.

OBJECT_USER = *OWN / *ALL / <var: char 1..8> / <c-string 1..8>
User ID of the owner of the object class instances for which the subscription is required.

*OWN	refers to the user ID of the task executing the program.
*ALL	requires a subscription for any user
<var: char 1..8>	requires a subscription for a particular user
<c-string 1..8>	requires a subscription for a particular user

OBJECT_NAME = <var: char 1..8> / <c-string 1..8>
Identification of the object class for which the subscription is required.

OBJECT_ID = <var: char 1..16> / <c-string 1..16>
Identification of the object class instance for which the subscription is required

DELIVERY_ADDR = <var: char 1..224> / <c-string 1..224>
Address the notification has to be sent to. The contents of this field depend on the requested delivery method

DELIVERY_METH = <var: char 1..8> / <c-string 1..8>
Notification delivery method that must be used for the notification

EVENTS_TYP = *ALL / *LIST
Type of the parameter EVENTS

*ALL default value . All events must be notified. The EVENTS parameter is ignored.

*LIST the EVENTS parameter contains a list of events that must be notified.

EVENTS = array(20) elem: <c-string 1..24> / <var: char 1..24>
List of up to 20 event names for which a notification is required.

ATTRIBUTES_TYP = *NONE / *ALL / *LIST
Type of the parameter ATTRIBUTES

*NONE default value. No specific attributes are required.
The ATTRIBUTES parameter is ignored

*ALL all the object instance attributes are required.
The ATTRIBUTES parameter is ignored

*LIST the ATTRIBUTES parameter contains a list of attributes

ATTRIBUTES = array(20) elem: <c-string 1..64> / <var: char 1..64>
List of attributes you want to find in the notification text. Up to 20 attribute names can be specified.

USER_DATA = <var: char 1..63>/ <c-string 1..63> / *NONE
Free text that will be included in the notification text. *NONE is the default value - no text will be included in the notification text.

COMMENTS = <var: char 1..200> / <c-string 1..200> / *NONE
Free text that will be only visible by a SHOW-NOTIFICATION-RESOURCES statement of the Notification Resources Manager. *NONE is the default value - no text will be visible.

SUB_MODE = *ACCEPT_NOTIFICATION / *REJECT_NOTIFICATION
Type of the subscription.

SUB_MODE = *ACCEPT_NOTIFICATION
*ACCEPT_NOTIFICATION indicates that the subscription is not a negative one.

SUB_MODE = *REJECT_NOTIFICATION

*REJECT_NOTIFICATION indicates that the subscription is a negative subscription i.e. that for the current object instance the user does not want to be notified at all. This also means that even if the user owns permanent subscriptions recorded in the notification file, he/she will not be notified at all about the object instance. The parameters EVENTS, ATTRIBUTES, USER_DATA are ignored in this case.

Return codes

Return code	Meaning
00 00 0000	Successful processing
00 01 0001	Invalid parameter list area
00 01 0002	Parameter value error
00 01 0003	Object not found
00 01 0004	Event not found
00 01 0005	Method not found
00 02 0006	Internal error while accessing object class resource
00 02 0007	Internal error while accessing event resource
00 02 0008	Internal error while accessing method resource
00 02 0009	Own user ID not found
00 02 000A	Internal error while recording the subscription

Structure layouts

SNPEVS (ASS)

For Assembler, the macro SNPEVS has the following structure layout:

```
*************************************************************************
*  BEGIN-INTERFACE   SNPEVS
*
*  TITLE             (/ SNS Event subscription interface /)
*  NAME              SNPEVS
*  DOMAIN            SNS
*  LANGUAGE          ASS
*  COPYRIGHT         (C) Fujitsu Siemens Computers GmbH 2003
*                        ALL RIGHTS RESERVED
*  COMPILATION-SCOPE USER
*  INTERFACE-TYPE    CALL
*  RUN-CONTEXT       TU
*
*  PURPOSE           (/ Event subscription interface parameter list /)
*
*  SYNTAX            (/ Syntax Variant 1:
*                           SNPEVS  MF = C|D|L|M
*                         , PREFIX     = [S] | <name>
*                         , MACID      = [EVS] | <name>
*                         , EQUATES    = [YES] | NO
*                         , OBJECT_USER= <var: char 1..8> |
*                                        <c-string 1..8> |
*                                        *OWN |
*                                        *ALL
*                         , OBJECT_NAME= <var: char 1..8> |
*                                        <c-string 1..8>
*                         , OBJECT_ID  = <var: char 1..16> |
*                                        <c-string 1..16>
*                         , DELIVERY_ADDR= <var: char 1..224> |
*                                          <c-string 1..224>
*                         , DELIVERY_METH= <var: char 1..8> |
*                                          <c-string 1..8>
*                         , EVENTS_TYP = *ALL |
*                                        *LIST |
*                                        default *ALL
*                         , EVENTS     = array(20)
*                                        elem: <c-string 1..24> |
*                                              <var: char 1..24>
*                         , ATTRIBUTES_TYP= *NONE |
*                                           *ALL |
*                                           *LIST |
*                                           default *NONE
```

```
*                          , ATTRIBUTES = array(20)
*                                         elem: <c-string 1..64> |
*                                               <var: char 1..64>
*                          , USER_DATA  = <var: char 1..63> |
*                                         <c-string 1..63> |
*                                         *NONE |
*                                         default *NONE
*                          , COMMENTS   = <var: char 1..200> |
*                                         <c-string 1..200> |
*                                         *NONE |
*                                         default *NONE
*                          , SUB_MODE   = *ACCEPT_NOTIFICATION |
*                                         *REJECT_NOTIFICATION |
*                                         default ACCEPT_NOTIFICATION
*                                                                  /)
*
*  REMARKS           (/ Welcome To Notification Service /)
*
*  END-INTERFACE     SNPEVS.
*********************************************************************
```

SNPEVS.H (C)

See Assembler description.

SNPEVSC (CPP)

See Assembler description.

Example for the use of SNPEVS

```
SLMWA    @PAR  D=YES,PLIST=(SLMPL@)
*
SNPREQT  SNPREQT MF=C
SNPEVS   SNPEVS  MF=C
*
*
SLMWA    @PAR  LEND=YES

         ...
         MVC     SNPEVS(SEVS#),SNPEVSL
         SNPEVS  MF=M,PARAM=SNPEVS,OBJECT_USER=*OWN, -
                 OBJECT_NAME=CSTOBJ,OBJECT_ID='ABCD', -
                 DELIVERY_ADDR=CSTADDR, -
                 DELIVERY_METH='MAILTO', -
                 EVENTS_TYP=*ALL,ATTRIBUTES_TYP=*NONE, -
                 USER_DATA=*NONE,COMMENTS='My new notification', -
                 SUB_MODE=*ACCEPT_NOTIFICATION
         MVC     SNPREQT(SREQ#),SNPREQTL
         SNPREQT MF=M,PARAM=SNPREQT,FCT_TYP=*RAISE_SUB, -
                 FCT_INTERFACE=A(SNPEVS)
         SNPREQT MF=E,PARAM=SNPREQT
         @IF     NZ
         OC      SREQFHDR,SREQFHDR
         @THEN                   , Error case
         ...
         @ELSE
         @IF     NZ
         CLI     SEVSSR1,0
         @THEN                   , Error case
         ...
         @BEND
         ...

CSTOBJ   DC      CL8'SPOOLJOB'
CSTADDR  DC      CL16'xxx@osl.be'
SNPREQTL SNPREQT MF=L
SNPEVSL  SNPEVS  MF=L
```

8.2 SNPREQT - Access to the Notification Service

User group: Nonprivileged user
Programming languages: Assembler, C, CPP
Macro type: S

This interface is the only one that allows the user and the system to call SNS. All the other API's are only parameter lists that must be anchored in the SNPREQT interface.

Format

Operation	Operands
SNPREQT	,MF = C / D / E / I / L / M
	,PREFIX = S / SNPREQT / <name>
	,MACID = <name 1..3>
	,ENTRY = YES / NO
	,PARAM = <name 1..27>
	,CALLER = USER / SYSTEM
	,FCT_TYPE = *RAISE_EV / *RAISE_SUB
	,FCT_INTERFACE = <var: pointer>

Description of the operands

MF = C / D / I / L / M
Type of the macro call. You will find more information in the “Executive Macros“ manual.
Possible values for the different programming languages:

Assembler:	C, D, E, L, M
C:	--
CPP:	D,E,I,M

Note:
You can only specify pointer variables (“var:pointer” operand value) with MF=M. In addition, a list must be generated with MF=D or MF=C.

PREFIX = S / SNPEVS / <name>
Specifies the first character of field names and equates.
Default values for the different programming languages:

Assembler:	S
C:	--
CPP:	SNPREQT

MACID = <name 1..3>
Specifies the second to fourth characters (inclusive) of the field names and equates.
Default values for the different programming languages:

Assembler:	REQ
C:	--
CPP:	--

ENTRY = YES / NO
Specifies whether the relevant entries are generated or not.
The default value is YES for CPP.

PARAM = <name 1..27>
Specifies the address of the operand list (permitted only in the case of MF formats 2 and 3).
You will find more information in the “Executive Macros” manual.

CALLER = USER / SYSTEM
Specifies if the macro is called by a user or by the system.

FCT_TYPE = *RAISE_EV / *RAISE_SUB
Type of the function that must be called and the parameter list of which is anchored by the operand FCT_INTERFACE.

*RAISE_EV	the raise event function is requested, the parameter list anchored must be a SNPREV parameter list
*RAISE_SUB	the subscription creation function is requested, the parameter list anchored must be a SNPEVS parameter list
all other values	all other values are reserved for internal use

FCT_INTERFACE = <var: pointer>
Address of the function parameter list

Return codes

System (TPR) return codes are returned in the standard header return code of the parameter list:

Return code	**Meaning**
00 00 0000	Successful processing
00 01 0000	Parameter error
00 20 0000	System error
00 40 0000	Validation error
00 00 FFFF	Linkage error (subsystem SNRTP not available)

User (TU) return codes are returned in the ISL command return code of the parameter list:

ISL command return code	**Meaning**
00 00 CMD0001	Successful processing
00 01 CMD0202	Parameter error
00 20 CMD0221	System error
00 40 SNP0801	Validation error
Std header return code	
00 00 FFFF	Linkage error (subsystem SNRTP not available)

Pay attention that even if the return code of the SNPREQT is ok, it is necessary to check the return code of the anchored parameter list. This one contains the return code of the requested function.

Structure layouts

SNPREQT (ASS)

For Assembler, the macro SNPREQT has the following structure layout:

```
************************************************************************
*  BEGIN-INTERFACE   SNPREQT
*
*  TITLE             (/ SNS SVC interface /)
*  NAME              SNPREQT
*  DOMAIN            SNS
*  LANGUAGE          ASS
*  COPYRIGHT         Fujitsu Siemens Computers GmbH 2003
*                        ALL RIGHTS RESERVED
*  COMPILATION-SCOPE USER
*  INTERFACE-TYPE    CALL
*  RUN-CONTEXT       TU
*
*  PURPOSE           (/ Interface parameter list for SNS SVC /)
*
*  SYNTAX            (/ Syntax Variant 1:
*                           SNPREQT  MF = C|D|L|M|E
*                         , PREFIX     = [S] | <name>
*                         , MACID      = [REQ] | <name>
*                         , PARAM      = <name 1..27>
*                         , CALLER     = <c-string_without_quotes 1..6>
*                         , EQUATES    = [YES] | NO
*                         , FCT_TYPE   = *NOTIFICATION |
*                                        *RAISE_EV |
*                                        *SNPDINI |
*                                        *SNPDRCV |
*                                        *SNPDRET |
*                                        *SNPDEND |
*                                        *RECOVERY |
*                                        *TIMESTAMP |
*                                        *RAISE_SUB |
*                                        default *NOTIFICATION
*                         , FCT_INTERFACE= <var: pointer> /)
*
*  REMARKS           (/ Welcome To Notification Service /)
*
*
*  END-INTERFACE     SNPREQT.
************************************************************************
```

SNPREQT.H (C)

See Assembler description.

SNPREQTC (CPP)

See Assembler description.

Example for the use of SNPREQT

See the other interface call examples.

8.3 SNPREV - Raise event interface

User group: Nonprivileged users
Programming languages: Assembler, C, CPP
Macro type: S

This interface describes the parameter list allowing the raise of an event from any BS2000 product or application. The call to SNS must be done by the SNPREQT interface.

Format

Operation	Operands
SNPREV	,MF = C / D / I / L / M
	,PREFIX = S / SNPREV / <name>
	,MACID = <name 1..3>
	,PARAM = <name 1..27>
	,EVENT_NAME = <var: char 1..24> / <c-string 1..24>
	,OBJECT_NAME = <var: char 1..8> / <c-string 1..8>
	,OBJECT_ID = <var: char 1..16> / <c-string 1..16>
	,OBJECT_USER = <var: char 1..8> / <c-string 1..8>
	,OBJ_INFO_ADDR = <var: pointer>
	,OBJ_INFO_LEN = <var: int 0..2147483647> / <integer 0..2147483647>
	,SUBS_NAME = <var: char 1..8> / <c-string 1..8>

Description of the operands

MF = C / D / I / L / M
Type of the macro call. You will find more information in the “Executive Macros“ manual.
Possible values for the different programming languages:

Assembler:	C, D, L, M
C:	--
CPP:	D,I,M

Note

You can only specify pointer variables (“var:pointer” operand value) with MF=M. In addition, a list must be generated with MF=D or MF=C.

PREFIX = S / SNPEVS / <name>
Specifies the first character of field names and equates.
Default values for the different programming languages:

Assembler:	S
C:	--
CPP:	SNPREV

MACID = <name 1..3>
Specifies the second to fourth characters (inclusive) of the field names and equates.
Default values for the different programming languages:

Assembler:	REV
C:	--
CPP:	--

PARAM = <name 1..27>
Specifies the address of the operand list (permitted only in the case of MF formats 2 and 3). You will find more information in the “Executive Macros” manual.

EVENT_NAME = <var: char 1..24> / <c-string 1..24>
Name of the event that is to be raised

OBJECT_NAME = <var: char 1..8> / <c-string 1..8>
Name of the object class the event is relative to

OBJECT_ID = <var: char 1..16> / <c-string 1..16>
Identification of the object class instance for which the event is raised

OBJECT_USER = <var: char 1..8> / <c-string 1..8>
User ID of the owner of the object class instance for which the event is raised

OBJ_INFO_ADDR = <var: pointer>
Address of a buffer containing the attributes and their values of the object class instance for which the event is raised. The format of the buffer contents must be a character string respecting the following format:

```
<attribute name1>=attribute value1,…., <attribute nameN>=attribute valueN
```

where

attribute name name of the object attribute (maximum 64 characters)

attribute value value of the attribute (maximum 256 characters)

OBJ_INFO_LEN = <var: int 0..2147483647> / <integer 0..2147483647>
Length of the object info buffer. The maximum length is 8184 characters.

SUBS_NAME = <var:char 1..8> / <c-string 1..8>
Name of the subsystem caller (e.g. SPOOL, RSO).

Return codes

Return code	Meaning
00 00 0000	Successful processing
01 00 0000	Warning - object class disable
02 00 0000	Warning - event disable
00 01 0005	Invalid parameter list
00 01 0006	Object class not found
00 01 0008	Event not found
00 32 0001	Request memory error
00 32 0002	Internal error while accessing object class resource
00 32 0003	Internal error while accessing event resource
00 32 0007	Sub TCB not available

Structure layouts

SNPREV (ASS)

For Assembler, the macro SNPREV has the following structure layout:

```
*************************************************************************
*  BEGIN-INTERFACE   SNPREV
*
*  TITLE             (/ SNS Raise-event interface /)
*  NAME              SNPREV
*  DOMAIN            SNS
*  LANGUAGE          ASS
*  COPYRIGHT         (C) Fujitsu Siemens Computers GmbH 2003
*                        ALL RIGHTS RESERVED
*  COMPILATION-SCOPE USER
*  INTERFACE-TYPE    CALL
*  RUN-CONTEXT       TU
*
*  PURPOSE           (/ Raise-event interface parameter list /)
*
*  SYNTAX            (/ Syntax Variant 1:
*                           SNPREV  MF = C|D|L|M
*                         , PREFIX     = [S] | <name>
*                         , MACID      = [REV] | <name>
*                         , EQUATES    = [YES] | NO
*                         , EVENT_NAME = <var: char 1..24> |
*                                        <c-string 1..24>
*                         , OBJECT_NAME= <var: char 1..8> |
*                                        <c-string 1..8>
*                         , OBJECT_ID  = <var: char 1..16> |
*                                        <c-string 1..16>
*                         , OBJECT_USER= <var: char 1..8> |
*                                        <c-string 1..8>
*                         , SUBS_NAME  = <var: char 1..8> |
*                                        <c-string 1..8>
*                         , OBJ_INFO_ADDR= <var: pointer>
*                         , OBJ_INFO_LEN= <var: int 0..2147483647> |
*                                         <integer 0..2147483647> /)
*                         , SUBS_NAME = <var: char 1..8> / <c-string 1..8>
*
*  REMARKS           (/ Welcome To Notification Service /)
*
*  END-INTERFACE     SNPREV.
*************************************************************************
```

SNPREV.H (C)

See Assembler description.

SNPREVC (CPP)

See Assembler description.

Example for the use of SNPREV

```
SLMWA    @PAR  D=YES,PLIST=(SLMPL@)
*
SNPREQT  SNPREQT MF=C
SNPREV   SNPREV  MF=C
*
SLMWA    @PAR  LEND=YES

         ...
         MVC     SNPREV(SREV#),SNPREVL
         SNPREV  MF=M,PARAM=SNPREV, -
                 EVENT_NAME=CSTEVT, -
                 OBJECT_NAME=CSTOBJ, -
                 OBJECT_USER='UID1', -
                 OBJECT_ID='ABCDOOOOD241ZEO8', -
                 OBJ_INFO_ADDR=A(OBJINF), -
                 OBJ_INFO_LEN=OBJINFL, -
                 SUBS_NAME='SPOOL'
         MVC     SNPREQT(SREQ#),SNPREQTL
         SNPREQT MF=M,PARAM=SNPREQT,FCT_TYP=*RAISE_EV, -
                 FCT_INTERFACE=A(SNPREV)
         SNPREQT MF=E,PARAM=SNPREQT
         @IF     NZ
         OC      SREQFHDR,SREQFHDR
         @THEN                   , Error case
         ...
         @ELSE
         @IF     NZ
         CLI     SREVSR1,0
         @THEN                   , Error case
         ...
         @BEND
         ...

CSTEVT   DC      CL24'PRINTJOBCOMPLETED'
CSTOBJ   DC      CL8'SPOOLJOB'
OBJINF   DC      CL38'<TSN>=ABCD,<ACCOUNT>=XYZ,<USERID>=UID1'
SNPREQTL SNPREQT MF=L
SNPREVL  SNPREV  MF=L
```

9 SNS messages

This section describes the SNS messages. All messages concerning SNS are classified according to the following conventions:

– The message class ID 'SNC' is reserved for all messages sent by the PROCEDURE notification delivery method

– The message class ID 'SNF' is reserved for all messages sent by the FILE notification delivery method

– The message class ID 'SNK' is reserved for all messages sent by the TU kernel subsystem

– The message class ID 'SNM' is reserved for all messages sent by the MAILTO notification delivery method

– The message class ID 'SNP' is reserved for all messages sent by the TPR subsystem

– The message class ID 'SNR' is reserved for all messages sent by the Notification Resources Manager program

– The message class ID 'SOM' is reserved for all messages sent by the OPGMAIL notification delivery method

SNC0001 Error '(&00)' from interface '(&01)'

Meaning
(&00) : return code from interface standard header
(&01) : system/internal interface name

SNC0003 Error when accessing template '(&01)' from library '(&00)'

Meaning
(&00) : library name
(&01) : template name
A error occurred when accessing the template located in the library

Response
Please check if the library exists or is not corrupted or check if the element(version,type) exists or is not corrupted

SNC0006 `Error when accessing template '(&00)'`

Meaning
(&00) : template name
An error occurred when accessing the template

Response
Please check if the template exists or is not corrupted

SNC0007 `Class '(&00)' memory request error`

Meaning
(&00) : memory class

SNC0008 `Warning : syntax error on record '(&00)' of the template '(&01)'.`

Meaning
(&00) : record number
(&01) : template name
A syntax error occurred when processing a record of the template. Processing continues.

Response
Please check the record in function of the syntax

SNC0009 `'(&01)' Error '(&00)' for pathname '(&02)'`

Meaning
(&00) : return code from interface standard header
(&01) : system/internal interface name
(&02) : filename/jvname

SNC9999 `MTHPROC trace: '(&00)'`

SNF0001 Error '(&00)' from interface '(&01)'

Meaning
(&00) : return code from interface standard header
(&01) : system/internal interface name

SNF0002 Notification filename '(&00)' given as recipient address is not valid

Meaning
(&00) : recipient address

SNF0003 Error when accessing template '(&01)' from library '(&00)'

Meaning
(&00) : library name
(&01) : template name
A error occurred when accessing the template located in the library

Response
Please check if the library exists or is not corrupted or check if the element(version,type) exists or is not corrupted

SNF0004 Backup of notification file '(&00)' not performed.

Meaning
(&00) : notification filename

SNF0006 Error when accessing template '(&00)'

Meaning
(&00) : template name
An error occurred when accessing the template

Response
Please check if the template exists or is not corrupted

SNF0007 Class '(&00)' memory request error

Meaning
(&00) : memory class

SNF0008 Job variable '(&00)' used by the method is in a inconsistent state.

Meaning
(&00) : job variable name

SNF0009 '(&01)' Error '(&00)' for pathname '(&02)'

Meaning
(&00) : return code from interface standard header
(&01) : system/internal interface name
(&02) : filename/jvname

SNF0100 Error '(&00)' from interface '(&01)'

Meaning
(&00) : main code from interface
(&01) : system/internal interface name

SNF0101 Recipient file is currently not active.

SNF0102 Backup file '(&00)' created

Meaning
(&00) : filename

SNF0103 Backup of recipient file not possible for the moment. Request is in delay.

Meaning
The recipient file is currently opened by at least one method.
Backup will be performed as soon as possible.

SNF0104 Backup of recipient file already in progress. Request is rejected.

SNF9999 MTHFILE trace: '(&00)'

SNKCOPY Copyright (C) '(&00)' '(&01)' All Rights Reserved

SNKLOAD Program '(&00)',Version '(&01)' of '(&02)' loaded

SNK0001 Error '(&00)' from interface '(&01)'

Meaning
(&00) : return code from interface standard header
(&01) : system/internal interface name

SNK0002 Connection to Notification Manager task failed

Meaning
An error occurred during the connection with SNPG task

SNK0003 Connection to Notification Manager task successfully established

SNK0004 Connection to Notification Manager task closed

SNK0005 Error while processing event '(&00)'-'(&01)'-'(&02)'. No notification has been sent

SNK0006 Error '(&00)' from interface '(&01)' when loading notification method '(&02))'

Meaning
(&00) : return code from interface standard header
(&01) : system/internal interface name
(&02) : method name

SNK0007 Error '(&00)' from interface '(&01)' when unloading notification method '(&02))'

Meaning
(&00) : return code from interface standard header
(&01) : system/internal interface name
(&02) : method name

SNK0008 Load of notification method '(&00)' successfully processed

SNK0009 An error occurred during the execution of the method '(&00)'. This method has been disabled

Meaning
(&00) = method name
An unrecoverable error occurred during the execution of the method. The method has been disabled automatically. There will be no notification raised for the subscriptions requiring the method.

Response
Correct the method, then start the Notification Manager utility to enable the method again

SNK9999 SNRKERN trace: '(&00)'

SNM0001 Error '(&00)' from interface '(&01)'

Meaning
(&00) : return code from interface standard header
(&01) : system/internal interface name

SNM0002 Batch with jobname '(&00)' abnormally terminated or has been cancelled.

Meaning
(&00) : jobname of the batch

SNM0003 Error when accessing template '(&01)' from library '(&00)'

Meaning
(&00) : library name
(&01) : template name
A error occurred when accessing the template located in the library

Response
Please check if the library exists or is not corrupted or check if the element(version,type) exists or is not corrupted

SNM0004 InterNet Value error when processing mail file '(&00)'. See trace file '(&01)' for more information

Meaning
(&00) : mail filename
(&01) : trace filename
An interNet Value error occurred when processing the mail file to send

Response
Please check in the associated trace file the reason of the error and correct it

SNM0005 InterNet Value warning when processing mail file '(&00)'. See trace file '(&01)' for more information

Meaning
(&00) : mail filename
(&01) : trace filename
An interNet Value warning occurred when processing the mail file to send

Response
Please check in the associated trace file the reason of the warning and correct it

SNM0009 '(&01)' Error '(&00)' for pathname '(&02)'

Meaning
(&00) : return code from interface standard header
(&01) : system/internal interface name
(&02) : filename/jvname

SNM9999 MTHMAIL trace: '(&00)'

SNP0001 Error '(&00)' from interface '(&01)'

Meaning
(&00) : return code from interface standard header
(&01) : system/internal interface name

SNP0002 Raise Event '(&00)' for object class '(&01)' failed. Reason: '(&02)'

Meaning
(&00) : Event name
(&01) : Object Class name
(&02) : Reason of the error:
- 01: Input parameter list not valid
- 02: Object information area not valid
- 03: Internal error during the get of Object Class resource
- 04: Object Class not found
- 05: Internal error during the get of Event resource
- 06: Event not found
- 07: Get memory error
- 08: SNPG system task access error

SNP0003 Add resource failed. Reason: '(&00)'

Meaning
(&00) : Reason of the error:
- 01: Get memory error
- 02: SNPG task access error
- 03: SNPG internal error

SNP0004 Modify resource failed. Reason: '(&00)'

Meaning
(&00) : Reason of the error:
- 01: Get memory error
- 02: SNPG task access error
- 03: SNPG internal error

SNP0005 Get resource failed. Reason: '(&00)'

Meaning
(&00) : Reason of the error:
- 01: Get count internal error
- 02: Get memory error
- 03: Get resource internal error

SNP0006 Remove resource failed. Reason: '(&00)'

Meaning
(&00) : Reason of the error:
01: Get memory error
02: SNPG task access error
03: SNPG internal error

SNP0007 Internal error during cache memory initialisation

Meaning
Class 4 memory for SNRTP resources is not successfully initialized.

SNP0008 Copying the '(&01)' file into '(&02)' failed. DMS error: '(&00)'

Meaning
An error occurred when creating the NOTIFICATION.PARAMETERS.copy file.
Help of the DMS error code provides additional information

SNP0009 Add resource '(&00)' into the '(&02)' file failed. DMS Error: '(&01)'

Meaning
An error occurred while adding a new record in the NOTIFICATION.PARAMETERS file.
(&00) : Key of the record
(&01) : DMS error
(&02) : Notification parameter file

SNP0010 Modify resource '(&00)' in the '(&03)' file failed. '(&01)' operation failed with DMS Error: '(&02)'

Meaning
An error occurred while adding a new record in the NOTIFICATION.PARAMETERS file.
(&00) : Key of the record
(&01) : DMS operation
(&02) : DMS error
(&03) : Notification parameter file

SNP0011 Initialisation of cache memory failed during access to the '(&01)' file. DMS Error: '(&00)'

Meaning
An error occurred while reading the NOTIFICATION.PARAMETERS file.
(&00) : DMS error
(&01) : Notification parameter file

SNP0012 Initialisation of cache memory failed during data insert

SNP0013 Add notification structure '(&00)' in cache memory failed. Error: '(&01)'

Meaning
(&00) : Structure key
(&01) : Error code

SNP0014 Remove resource '(&00)' from the '(&02)' file failed. DMS Error: '(&01)'

Meaning
An error occurred while removing a record from the NOTIFICATION.PARAMETERS file.
(&00) : Key of the record
(&01) : DMS error
(&02) : Notification parameter file

SNP0015 Remove notification structure '(&00)' from cache memory failed. Error: '(&01)'

Meaning
(&00) : Structure key
(&01) : Error code

SNP0016 Open '(&02)' file processing failed during operation '(&01)'. Error code: '(&00)'

Meaning
An error occurred during the open processing of the NOTIFICATION.PARAMETERS file.
(&00) : Error code
(&01) : Operation
*REQ = Request FCB
*FSTAT = Fstat of NOTIFICATION.PARAMETERS file
*FSTAT2 = Fstat of NOTIFICATION.PARAMETERS.COPY file
*COPY = Copy of NOTIFICATION.PARAMETERS.COPY file
*CREATE = Create the NOTIFICATION.PARAMETERS file
*OPEN = Open the NOTIFICATION.PARAMETERS file
(&02) : Notification parameter file

SNP0017 Error '(&00)' from interface '(&01)' when loading check privilege module '(&02))'

Meaning
(&00) : return code from interface standard header
(&01) : system/internal interface name
(&02) : library name(object module name)

SNP0018 Error '(&00)' from interface '(&01)' when unloading check privilege module '(&02))'

Meaning
(&00) : return code from interface standard header
(&01) : system/internal interface name
(&02) : object module name

SNP0019 subsystem can not be created. Open $TSOS.SNRTP.BATCH.PROC file processing failed during operation '(&01)'. Error code: '(&00)'

Meaning
An error occurred during the open processing of the SNRTP.BATCH.PROC file.
(&00) : Error code
(&01) : Operation
*REQ = Request FCB
*FSTAT = Fstat of $TSOS.SNRTP.BATCH.PROC file
*CREATE = Create the $TSOS.SNRTP.BATCH.PROC file
*OPEN = Open the $TSOS.SNRTP.BATCH.PROC file

SNP0020 An error has been detected by SNPG task during a notification processing. SNPG task is waiting SNRTP shutdown

Meaning
An unrecoverable error has stopped the processing of the SNPG task. See previous message for more explanation.

SNP0021 Access to Notification Manager task temporarily unavailable

Meaning
The SNPG task is currently not available. Retry command later.

SNP0022 Access to Notification Manager task no longer available

Meaning
The SNPG task is out of order.A restart of SNRTP subsystem is necessary.
Stop the program and retry later.

SNP0023 Error '(&00)' while accessing library '(&01)'. In system mode /HELP-MSG PLA'(&00)'

Meaning
(&00) = PLAM error number
(&01) = Library name
An error occurred during the library access

SNP0024 Connection error '(&00)' from interface '$ESMCONI' while accessing privilege policy '(&01)' of subsystem '(&02). No notification launched

Meaning
(&00) = $ESNCONI error number
(&01) = Policy name = object class domain
(&02) = Subsystem name
An error occurred during the connection to the privilege policy (&01) for the subsystem (&02). No notification will be launched for the Object Classes belonging to the domain (&01)

SNP0100 '(&00)' subsystem can not be created. No string operand possible

Meaning
(&00) : subsystem name
The operand STRING is not supported for this subsystem

Response
Start the subsystem without the STRING operand

SNP0101 '(&00)' subsystem can not be created. Invalid specification in '(&01)' parameter file

Meaning
(&00) : subsystem name
(&01) : subsystem parameter file
The (&01) file does not exist or a bad information has been introduced in the file.

Response
Correct or restore the (&01) file and retry startup

SNP0102 '(&02)' subsystem can not be created. DMS error '(&00)' during processing 'GET','FILE' OR 'OPEN' macro for file '(&01)'. In system mode: /HELP-MSG DMS'(&00)'

Meaning
(&00) : DMS error code
(&01) : subsystem parameter file
(&02) : subsystem name
The (&01) file does not exist or a bad information has been introduced in the file.

Response
Correct or restore the (&01) file and retry startup

SNP0103 Class '(&00)' memory request error

Meaning
(&00) : memory class

SNP0104 Class '(&00)' memory release error

Meaning
(&00) : memory class

SNP0105 File '(&00)' used as notification parameters file

Meaning
(&00) : file name

SNP0106 File '(&00)' used as check privilege library

Meaning
(&00) : file name

SNP0107 File '(&00)' used as syslnk library for TU subsystem

Meaning
(&00) : File name

SNP0108 '(&00)' subsystem can not be created. DSSM return code '(&01)' from interface '(&02)'

Meaning
(&00) : subsystem name
(&01) : return code
(&02) : DSSM interface name

SNP0109 '(&00)' subsystem can not be created. '(&01)' task can not be created

Meaning
(&00) : subsystem name.
(&01) : tsn of the task.

SNP0110 Initialisation of communication with notification task '(&00)' failed

Meaning
(&00) : tsn of the TU notification task

SNP0111 Initialisation of communication with notification task '(&00)' successfully processed

Meaning
(&00) : tsn of the TU notification task

SNP0112 HOLD-SUBSYSTEM command for 'SNRTP' subsystem is not supported. Command has been rejected

Meaning
SNRTP subsystem does not support the HOLD subsystem feature

Response
In order to stop the SNRTP subsystem, please use /STOP-SUBYSTEM SNRTP

SNP0113 '(&00)' subsystem can not be stopped. DSSM return code '(&01)' from interface '(&02)'

Meaning
(&00) : subsystem name.
(&01) : return code
(&02) : DSSM interface name

SNP0114 '(&00)' subsystem can not be created. Userid '(&01)' does not exist

Meaning
(&00) : subsystem name
(&01) : notification userid
The (&01) userid does not exist

Response
Create the userid for the notification service and retry startup

SNP0801 Logical validation problem during command processing. Command rejected

SNP1000 The '(&00)' file does not exist. Do you want to restart from the '(&01)'? (Y=Yes, N=No)

Meaning
The NOTIFICATION.PARAMETERS file does not exist but the file NOTIFICATION.PARAMETERS.COPY exists. You can restart from this file.

Response
Answer Y if you want to restart from the .COPY file
Answer N otherwise. In this case the START-SUBSYSTEM SNRTP will be aborted

SNP1001 The TU SNRKERN task can not be created. Notification processing unavailable

Meaning
The batch task has not been created. Maybe due to job class not available. Check the job class, stop and restart the SNRTP subsystem.

```
SNRCOPY    Copyright (C) '(&00)' '(&01)' All Rights Reserved
SNRLOAD    Program '(&00)',Version '(&01)' of '(&02)' loaded
SNR0001    Processing completed.
SNR0002    Processing abnormally terminated.
SNR0003    System Notification Service not present.
```

Meaning
The SNRTP subsystem is not loaded.

Response
Please contact the system administrator.

```
SNR0004    SDF not present. Program aborted.
```

Meaning
The SDF subsystem is not loaded.

Response
Please contact the system administrator.

```
SNR0005    Program not defined in SDF. Program aborted.
```

Response
Please contact the system administrator.

```
SNR0006    Syntax file can not be found. Program aborted.
```

Response
Please contact the system administrator.

```
SNR0007    Error in XHCS during statement input. Program aborted.
```

Response
Please contact the system administrator.

```
SNR0008    Internal error '(&00)'.
```

Meaning
(&00): internal error code :

- x'01' : transfer area too small.
- x'02' : unknown statement.
- x'03' : unknown Sdf return code.
- x'04' : resource processing error.
- x'05' : unknown SNS return code.
- x'06' : Internal interface parameter list not correct.

Response
Please contact the system administrator.

SNR0009 '(&00)' object(s) removed.

Meaning
(&00): number of objects

SNR0010 No '(&00)' resource(s) found.

Meaning
(&00): Resource type

SNR0011 '(&00)' resource with name '(&01)' already exists.

Meaning
(&00): Resource type
(&01): Resource name

SNR0012 System interface '(&00)' error.

Meaning
(&00): Interface name

Response
Please contact the system administrator.

SNR0013 Not enough privilege to perform the action. Statement rejected.

SNR0014 Resource referenced by others objects. Statement rejected.

SNR0015 Statement interrupted by user.

SNR0016 Referenced '(&00)' resource with name '(&01)' doesn't exist.

Meaning
(&00): Resource type
(&01): Resource name

SOM0001 Error '(&00)' from interface '(&01)'

Meaning
(&00) : return code from interface standard header
(&01) : system/internal interface name

SOM0003 Error when accessing template '(&01)' from library '(&00)'

Meaning
(&00) : library name
(&01) : template name
An error occurred when accessing the template located in the library

Response
Please check if the library exists or is not corrupted or check if the element(version,type) exists or is not corrupted

SOM0004 OPG MAil error when processing mail file '(&00)'. See trace file '(&01)' for more information

Meaning
(&00) : mail filename
(&01) : trace filename
An OPG Mail error occurred when processing the mail file to send

Response
Please check in the associated trace file the reason of the error and correct it

SOM0005 OPG Mail warning when processing mail file '(&00)'. See trace file '(&01)' for more information

Meaning
(&00) : mail filename
(&01) : trace filename
An OPG Mail warning occurred when processing the mail file to send

Response
Please check in the associated trace file the reason of the warning and correct it

SOM0006 Access error when processing mail file '(&00)'. No mail send

Meaning
(&00) : mail filename
An error occurred when processing the mail file to send

Response
Please, check the content of the mail file

SOM0007 Error when accessing template '(&00)'

Meaning
(&00) : template name
An error occurred when accessing the template

Response
Please check if the template exists or is not corrupted

SOM0009 '(&01)' Error '(&00)' for pathname '(&02)'

Meaning
(&00) : return code from interface standard header
(&01) : system/internal interface name
(&02) : filename/jvname

SOM9999 OPGMAIL trace: '(&00)'

10 Appendix

SDF syntax representation

The following example shows the representation of the syntax of a command in a manual. The command format consists of a field with the command name. All operands with their legal values are then listed. Operand values which introduce structures and the operands dependent on these operands are listed separately.

HELP-SDF Alias: **HPSD**

GUIDANCE**-MODE** = ***NO** / ***Y**ES

,**SDF-COM**MANDS = ***NO** / ***Y**ES

,**ABBR**EVIATION**-RULES** = ***NO** / ***Y**ES

,**GUID**ED**-DIA**LOG = ***YES** (...)

***Y**ES(...)

SCREEN-STEPS = ***NO** / ***Y**ES

,**SPEC**IAL**-FUNC**TIONS = ***NO** / ***Y**ES

,**FUNC**TION**-KEYS** = ***NO** / ***Y**ES

,**NEXT-FIELD** = ***NO** / ***Y**ES

,**UNGUID**ED**-DIA**LOG = ***YES** (...) / ***NO**

***Y**ES(...)

SPECIAL**-FUNC**TIONS = ***NO** / ***Y**ES

,**FUNC**TION**-KEYS** = ***NO** / ***Y**ES

This syntax description is valid for SDF V4.5A.The syntax of the SDF command/statement language is explained in the following three tables.

Table 1: Notational conventions

The meanings of the special characters and the notation used to describe command and statement formats are explained in table 1.

Table 2: Data types

Variable operand values are represented in SDF by data types. Each data type represents a specific set of values. The number of data types is limited to those described in table 2.

The description of the data types is valid for the entire set of commands/statements. Therefore only deviations (if any) from the attributes described here are explained in the relevant operand descriptions.

Table 3: Suffixes for data types

Data type suffixes define additional rules for data type input. They contain a length or interval specification and can be used to limit the set of values (suffix begins with *without*), extend it (suffix begins with *with*), or declare a particular task mandatory (suffix begins with *mandatory*). The following short forms are used in this manual for data type suffixes:

cat-id	cat
completion	compl
correction-state	corr
generation	gen
lower-case	low
manual-release	man
odd-possible	odd
path-completion	path-compl
separators	sep
temporary-file	temp-file
underscore	under
user-id	user
version	vers
wildcard-constr	wild-constr
wildcards	wild

The description of the ‘integer’ data type in table 3 contains a number of items in italics; the italics are not part of the syntax and are only used to make the table easier to read.
For special data types that are checked by the implementation, the table 3 contains suffixes printed in italics (see the *special* suffix) which are not part of the syntax.

The description of the data type suffixes is valid for the entire set of commands/statements. Therefore only deviations (if any) from the attributes described here are explained in the relevant operand descriptions.

Metasyntax

Representation	Meaning	Examples
UPPERCASE LETTERS	Uppercase letters denote keywords (command, statement or operand names, keyword values) and constant operand values. Keyword values begin with *.	**HELP-SDF** **SCREEN-STEPS** = ***NO**
UPPERCASE LETTERS in boldface	Uppercase letters printed in boldface denote guaranteed or suggested abbreviations of keywords.	**GUID**ANCE**-MODE** = ***Y**ES
=	The equals sign connects an operand name with the associated operand values.	**GUID**ANCE**-MODE** = ***NO**
< >	Angle brackets denote variables whose range of values is described by data types and suffixes (see tables 2 and 3).	**SYNTAX-F**ILE = <filename 1..54>
Underscoring	Underscoring denotes the default value of an operand.	**GUID**ANCE**-MODE** = ***NO**
/	A slash serves to separate alternative operand values.	**NEXT-FIELD** = ***NO** / ***Y**ES
(...)	Parentheses denote operand values that initiate a structure.	,**UNGUID**ED**-DIA**LOG = ***Y**ES (...) / ***NO**
[ ]	Square brackets denote operand values which introduce a structure and are optional. The subsequent structure can be specified without the initiating operand value.	**SELECT** = [***BY-ATTR**IBUTES](...)
Indentation	Indentation indicates that the operand is dependent on a higher-ranking operand.	,**GUID**ED**-DIA**LOG = ***Y**ES (...) ***Y**ES(...) **SCREEN-STEPS** = ***NO** / ***Y**ES

Table 1: Metasyntax (part 1 of 2)

Representation	Meaning	Examples
|	A vertical bar identifies related operands within a structure. Its length marks the beginning and end of a structure. A structure may contain further structures. The number of vertical bars preceding an operand corresponds to the depth of the structure.	**SUP**PORT = ***TAPE**(...) ***TAPE(...)** | **VOL**UME = ***ANY**(...) | ***ANY**(...) | | ...
,	A comma precedes further operands at the same structure level.	**GUID**ANCE-**MODE** = ***NO** / ***Y**ES ,**SDF-COM**MANDS = ***NO** / ***Y**ES
list-poss(n):	The entry “list-poss” signifies that a list of operand values can be given at this point. If (n) is present, it means that the list must not have more than n elements. A list of more than one element must be enclosed in parentheses.	list-poss: ***SAM** / ***ISAM** list-poss(40): <structured-name 1..30> list-poss(256): ***OMF** / ***SYSLST**(...) / <filename 1..54>
Alias:	The name that follows represents a guaranteed alias (abbreviation) for the command or statement name.	**HELP-SDF** Alias: **HPSDF**

Table 1: Metasyntax (part 2 of 2)

Data types

Data type	Character set	Special rules
alphanum-name	A…Z 0…9 $, #, @	
cat-id	A…Z 0…9	Not more than 4 characters; must not begin with the string PUB
command-rest	freely selectable	
composed-name	A…Z 0…9 $, #, @ hyphen period catalog ID	Alphanumeric string that can be split into multiple substrings by means of a period or hyphen. If a file name can also be specified, the string may begin with a catalog ID in the form :cat: (see data type filename).
c-string	EBCDIC character	Must be enclosed within single quotes; the letter C may be prefixed; any single quotes occurring within the string must be entered twice.
date	0…9 Structure identifier: hyphen	Input format: yyyy-mm-dd jjjj: year; optionally 2 or 4 digits mm: month tt: day
device	A…Z 0…9 hyphen	Character string, max. 8 characters in length, corresponding to a device available in the system. In guided dialog, SDF displays the valid operand values. For notes on possible devices, see the relevant operand description.
fixed	+, - 0…9 period	Input format: [sign][digits].[digits] [sign]: + or - [digits]: 0...9 must contain at least one digit, but may contain up to 10 characters (0...9, period) apart from the sign.

Table 2: Data types (part 1 of 6)

Data type	Character set	Special rules
filename	A…Z 0…9 $, #, @ hyphen period	Input format: [:cat:][$user.] { file | file(no) | group | group { (*abs) | (+rel) | (-rel) } } :cat: optional entry of the catalog identifier; character set limited to A...Z and 0...9; maximum of 4 characters; must be enclosed in colons; default value is the catalog identifier assigned to the user ID, as specified in the user catalog. $user. optional entry of the user ID; character set is A…Z, 0…9, $, #, @; maximum of 8 characters; first character cannot be a digit; $ and period are mandatory; default value is the user's own ID. $. (special case) system default ID file file or job variable name; may be split into a number of partial names using a period as a delimiter: $name_1$[.$name_2$[...]] $name_i$ does not contain a period and must not begin or end with a hyphen; file can have a maximum length of 41 characters; it must not begin with a $ and must include at least one character from the range A...Z.

Table 2: Data types (part 2 of 6)

Data type	Character set	Special rules
filename (contd.)		#file (special case) @file (special case) # or @ used as the first character indicates temporary files or job variables, depending on system generation. file(no) tape file name no: version number; character set is A...Z, 0...9, $, #, @. Parentheses must be specified. group name of a file generation group (character set: as for “file”) group{(*abs) (+rel) (-rel)} (*abs) absolute generation number (1-9999); * and parentheses must be specified. (+rel) (-rel) relative generation number (0-99); sign and parentheses must be specified.
integer	0…9, +, -	+ or -, if specified, must be the first character.
name	A…Z 0…9 $, #, @	Must not begin with 0...9.

Table 2: Data types (part 3 of 6)

Data type	Character set	Special rules
partial-filename	A…Z 0…9 $, #, @ hyphen period	Input format: [:cat:][$user.][partname.] :cat: see filename $user. see filename partname optional entry of the initial part of a name common to a number of files or file generation groups in the form: $name_1$.[$name_2$.[...]] $name_i$ (see filename). The final character of “partname” must be a period. At least one of the parts :cat:, $user. or partname must be specified.
posix-filename	A...Z 0...9 special characters	String with a length of up to 255 characters; consists of either one or two periods or of alpha-numeric characters and special characters. The special characters must be escaped with a preceding \ (backslash); the / is not allowed. Must be enclosed within single quotes if alter-native data types are permitted, separators are used, or the first character is a ?, ! or ^. A distinction is made between uppercase and lowercase.
posix-pathname	A...Z 0...9 special characters structure identifier: slash	Input format: [/]$part_1$/.../$part_n$ where $part_i$ is a posix-filename; max. 1023 characters; must be enclosed within single quotes if alter-native data types are permitted, separators are used, or the first character is a ?, ! or ^.

Table 2: Data types (part 4 of 6)

Data type	Character set	Special rules
product-version	A…Z 0…9 period single quote	Input format: [[C]'][V][m]m.naso['] correction status release status where m, n, s and o are all digits and a is a letter. Whether the release and/or correction status may/must be specified depends on the suffixes to the data type (see suffixes without-corr, without-man, mandatory-man and mandatory-corr in table 3). product-version may be enclosed within single quotes (possibly with a preceding C). The specification of the version may begin with the letter V.
structured-name	A…Z 0…9 $, #, @ hyphen	Alphanumeric string which may comprise a number of substrings separated by a hyphen. First character: A...Z or $, #, @
text	freely selectable	For the input format, see the relevant operand descriptions.
time	0…9 structure identifier: colon	Time-of-day entry: Input format: { hh:mm:ss / hh:mm / hh } hh: hours mm: minutes ss: seconds } Leading zeros may be omitted
vsn	a) A…Z 0…9 b) A…Z 0…9 $, #, @	a) Input format: pvsid.sequence-no max. 6 characters pvsid: 2-4 characters; PUB must not be entered sequence-no: 1-3 characters b) Max. 6 characters; PUB may be prefixed, but must not be followed by $, #, @.

Table 2: Data types (part 5 of 6)

Data type	Character set	Special rules
x-string	Hexadecimal: 00…FF	Must be enclosed in single quotes; must be prefixed by the letter X. There may be an odd number of characters.
x-text	Hexadecimal: 00…FF	Must not be enclosed in single quotes; the letter X must not be prefixed. There may be an odd number of characters.

Table 2: Data types (part 6 of 6)

Suffixes for data types

Suffix	Meaning
x..y *unit*	With data type “integer”: interval specification x minimum value permitted for “integer”. x is an (optionally signed) integer. y maximum value permitted for “integer”. y is an (optionally signed) integer. *unit* with “integer” only: additional units. The following units may be specified: *days* *byte* *hours* *2Kbyte* *minutes* *4Kbyte* *seconds* *Mbyte* *milliseconds*
x..y *special*	With the other data types: length specification For data types catid, date, device, product-version, time and vsn the length specification is not displayed. x minimum length for the operand value; x is an integer. y maximum length for the operand value; y is an integer. x=y the length of the operand value must be precisely x. *special* Specification of a suffix for describing a special data type that is checked by the implementation. “special” can be preceded by other suffixes. The following specifications are used: *arithm-expr* arithmetic expression (SDF-P) *bool-expr* logical expression (SDF-P) *string-expr* string expression (SDF-P) *expr* freely selectable expression (SDF-P) *cond-expr* conditional expression (JV) *symbol* CSECT or entry name (BLS)
with	Extends the specification options for a data type.
-compl	When specifying the data type “date”, SDF expands two-digit year specifications in the form yy-mm-dd to: 20jj-mm-tt if jj < 60 19jj-mm-tt if jj ≥ 60
-low	Uppercase and lowercase letters are differentiated.
-path-compl	For specifications for the data type “filename”, SDF adds the catalog and/or user ID if these have not been specified.

Table 3: Data type suffixes (part 1 of 7)

<table>
<tr><th>Suffix</th><th>Meaning</th></tr>
<tr><td>with (contd.)</td><td></td></tr>
<tr><td>-under</td><td>Permits underscores (_) for the data type “name”.</td></tr>
<tr><td>-wild(n)</td><td>Parts of names may be replaced by the following wildcards.
n denotes the maximum input length when using wildcards.
Due to the introduction of the data types posix-filename and posix-pathname, SDF now accepts wildcards from the UNIX world (referred to below as POSIX wildcards) in addition to the usual BS2000 wildcards. However, as not all commands support POSIX wildcards, their use for data types other than posix-filename and posix-pathname can lead to semantic errors.
Only POSIX wildcards or only BS2000 wildcards should be used within a search pattern. Only POSIX wildcards are allowed for the data types posix-filename and posix-pathname. If a pattern can be matched more than once in a string, the first match is used.
<table>
<tr><th>BS2000 wildcards</th><th>Meaning</th></tr>
<tr><td>*</td><td>Replaces an arbitrary (even empty) character string. If the string concerned starts with *, then the * must be entered twice in succession if it is followed by other characters and if the character string entered does not contain at least one other wildcard.</td></tr>
<tr><td>Terminating period</td><td>Partially-qualified entry of a name.
Corresponds implicitly to the string “./*”, i.e. at least one other character follows the period.</td></tr>
<tr><td>/</td><td>Replaces any single character.</td></tr>
<tr><td><s_x:s_y></td><td>Replaces a string that meets the following conditions:
– It is at least as long as the shortest string (s_x or s_y)
– It is not longer than the longest string (s_x or s_y)
– It lies between s_x and s_y in the alphabetic collating sequence; numbers are sorted after letters (A...Z, 0...9)
– s_x can also be an empty string (which is in the first position in the alphabetic collating sequence)
– s_y can also be an empty string, which in this position stands for the string with the highest possible code (contains only the characters X’FF’)</td></tr>
<tr><td><s_1,…></td><td>Replaces all strings that match any of the character combinations specified by s. s may also be an empty string. Any such string may also be a range specification “s_x:s_y” (see above).</td></tr>
</table>
</td></tr>
</table>

Table 3: Data type suffixes (part 2 of 7)

Suffix	**Meaning**	
with-wild(n) (contd.)	-s	Replaces all strings that do not match the specified string s. The minus sign may only appear at the beginning of string s. Within the data types filename or partial-filename the negated string -s can be used exactly once, i.e. -s can replace one of the three name components: cat, user or file.
	Wildcards are not permitted in generation and version specifications for file names. Only system administration may use wildcards in user IDs. Wildcards cannot be used to replace the delimiters in name components cat (colon) and user ($ and period).	
	POSIX wildcards	Meaning
	*	Replaces any single string (including an empty string). An * appearing at the first position must be duplicated if it is followed by other characters and if the entered string does not include at least one further wildcard.
	?	Replaces any single character; not permitted as the first character outside single quotes.
	[c_x-c_y]	Replaces any single character from the range defined by c_x and c_y, including the limits of the range. c_x and c_y must be normal characters.
	[s]	Replaces exactly one character from string s. The expressions [c_x-c_y] and [s] can be combined into [s_1c_x-c_ys_2].
	[!c_x-c_y]	Replaces exactly one character not in the range defined by c_x and c_y, including the limits of the range. c_x and c_y must be normal characters. The expressions [!c_x-c_y] and [!s] can be combined into [!s_1c_x-c_ys_2].
	[!s]	Replaces exactly one character not contained in string s. The expressions [!s] and [!c_x-c_y] can be combined into [!s_1c_x-c_ys_2].

Table 3: Data type suffixes (part 3 of 7)

<table>
<tr><th>Suffix</th><th>Meaning</th></tr>
<tr><td>with (contd.)</td><td></td></tr>
<tr><td>wild-
constr(n)</td><td>Specification of a constructor (string) that defines how new names are to be constructed from a previously specified selector (i.e. a selection string with wildcards). See also with-wild. n denotes the maximum input length when using wildcards.
The constructor may consist of constant strings and patterns. A pattern (character) is replaced by the string that was selected by the corresponding pattern in the selector.
The following wildcards may be used in constructors:
<table>
<tr><th>Wildcard</th><th>Meaning</th></tr>
<tr><td>*</td><td>Corresponds to the string selected by the wildcard * in the selector.</td></tr>
<tr><td>Termina-
ting period</td><td>Corresponds to the partially-qualified specification of a name in the selector;
corresponds to the string selected by the terminating period in the selector.</td></tr>
<tr><td>/ or ?</td><td>Corresponds to the character selected by the / or ? wildcard in the selector.</td></tr>
<tr><td><n></td><td>Corresponds to the string selected by the n-th wildcard in the selector, where n is an integer.</td></tr>
</table>
Allocation of wildcards to corresponding wildcards in the selector:
All wildcards in the selector are numbered from left to right in ascending order (global index).
Identical wildcards in the selector are additionally numbered from left to right in ascending order (wildcard-specific index).
Wildcards can be specified in the constructor by one of two mutually exclusive methods:
1. Wildcards can be specified via the global index: <n>
2. The same wildcard may be specified as in the selector; substitution occurs on the basis of the wildcard-specific index. For example:
the second “/” corresponds to the string selected by the second “/” in the selector</td></tr>
</table>

Table 3: Data type suffixes (part 4 of 7)

Suffix	Meaning
with-wild-constr(n) (contd.)	The following rules must be observed when specifying a constructor: – The constructor can only contain wildcards of the selector. – If the string selected by the wildcard <...> or [...] is to be used in the constructor, the index notation must be selected. – The index notation must be selected if the string identified by a wildcard in the selector is to be used more than once in the constructor. For example: if the selector “A/” is specified, the constructor “A<n><n>” must be specified instead of “A//”. – The wildcard * can also be an empty string. Note that if multiple asterisks appear in sequence (even with further wildcards), only the last asterisk can be a non-empty string, e.g. for “****” or “*//*”. – Valid names must be produced by the constructor. This must be taken into account when specifying both the constructor and the selector. – Depending on the constructor, identical names may be constructed from different names selected by the selector. For example: “A/*” selects the names “A1” and “A2”; the constructor “B*” generates the same new name “B” in both cases. To prevent this from occurring, all wildcards of the selector should be used at least once in the constructor. – If the constructor ends with a period, the selector must also end with a period. The string selected by the period at the end of the selector cannot be specified by the global index in the constructor specification.

Table 3: Data type suffixes (part 5 of 7)

<table>
<tr><th>Suffix</th><th>Meaning</th></tr>
<tr><td>with-wild-
constr(n)
(contd.)</td><td>Examples:
<table>
<tr><th>Selector</th><th>Selection</th><th>Constructor</th><th>New name</th></tr>
<tr><td>A//*</td><td>AB1
AB2
A.B.C</td><td>D<3><2></td><td>D1
D2
D.CB</td></tr>
<tr><td>C.<A:C>/<D,F></td><td>C.AAD
C.ABD
C.BAF
C.BBF</td><td>G.<1>.<3>.XY<2></td><td>G.A.D.XYA
G.A.D.XYB
G.B.F.XYA
G.B.F.XYB</td></tr>
<tr><td>C.<A:C>/<D,F></td><td>C.AAD
C.ABD
C.BAF
C.BBF</td><td>G.<1>.<2>.XY<2></td><td>G.A.A.XYA
G.A.B.XYB
G.B.A.XYA
G.B.B.XYB</td></tr>
<tr><td>A//B</td><td>ACDB
ACEB
AC.B
A.CB</td><td>G/XY/</td><td>GCXYD
GCXYE
GCXY. [1]
G.XYC</td></tr>
</table>
[1] The period at the end of the name may violate naming conventions (e.g. for fully-qualified file names).</td></tr>
<tr><td>without</td><td>Restricts the specification options for a data type.</td></tr>
<tr><td>-cat</td><td>Specification of a catalog ID is not permitted.</td></tr>
<tr><td>-corr</td><td>Input format: [[C]’][V][m]m.na[’]
Specifications for the data type product-version must not include the correction status.</td></tr>
<tr><td>-gen</td><td>Specification of a file generation or file generation group is not permitted.</td></tr>
<tr><td>-man</td><td>Input format: [[C]’][V][m]m.n[’]
Specifications for the data type product-version must not include either release or correction status.</td></tr>
<tr><td>-odd</td><td>The data type x-text permits only an even number of characters.</td></tr>
<tr><td>-sep</td><td>With the data type “text”, specification of the following separators is not permitted: ; = ( ) < > ␣ (i.e. semicolon, equals sign, left and right parentheses, greater than, less than, and blank).</td></tr>
<tr><td>-temp-
file</td><td>Specification of a temporary file is not permitted (see #file or @file under filename).</td></tr>
</table>

Table 3: Data type suffixes (part 6 of 7)

Suffix	Meaning
without (contd.)	
-user	Specification of a user ID is not permitted.
-vers	Specification of the version (see “file(no)”) is not permitted for tape files.
-wild	The file types posix-filename and posix-pathname must not contain a pattern (character).
mandatory	Certain specifications are necessary for a data type.
-corr	Input format: [[C]’][V][m]m.naso[’] Specifications for the data type product-version must include the correction status and therefore also the release status.
-man	Input format: [[C]’][V][m]m.na[so][’] Specifications for the data type product-version must include the release status. Specification of the correction status is optional if this is not prohibited by the use of the suffix without-corr.
-quotes	Specifications for the data types posix-filename and posix-pathname must be enclosed in single quotes.

Table 3: Data type suffixes (part 7 of 7)

Glossary

configuration file
File element (type D) included in the SYSLIB.SNRTP.010.METHOD library that contains the configuration parameters for a notification delivery method.

delivery method
see notification delivery method.

event
Notification resource. An event defines a change of state (either expected or unexpected) of an occurrence of an object class. An event always refers to an object class.

method
see notification delivery method.

notification delivery method
Notification resource. This is the mechanism by which SNS delivers event notifications. SNS sees the delivery methods as separate components that have to be plugged in. SNS V1.0B is released with different notification delivery methods (MAILTO, OPGMAIL, PROCEDURE, FILE).

notification resources
In SNS, the following resources exist: object class, event, notification delivery method, and subscription
All resources are stored in one central file, the notification resources file $SYSSNS.NOTIFICATION.PARAMETERS, and are managed by the Notification Resources Manager program.

notification resources file
Central file, in which all notification resources are stored. This file has the name $SYSSNS.NOTIFICATION.PARAMETERS and is managed with the Notification Resources Manager program.

Notification Resources Manager

The Notification Resources Manager (SNRMGR) is a program including dedicated functions for the management of notification resources and the customisation of the notification system definition. It can be started with the command START-NOTIFICATION-MANAGER.

object class

Notification resource. These are the objects which a user can be notified of.

privilege policy

If the default privilege policy provided by SNS is not sufficient, a specific privilege policy dedicated to the product can be implemented. The role of this privilege policy is to determine whether a subscription can generate a notification for the object instance that raised the event.

raise event

Occurrence of an event that causes some kind of processing informing SNS of the occurrence of the event.

registration

Definition of the notification resources in the notification resources file.

resource

see notification resources.

SNPCHKP

Privilege policy interface. The macro SNPCHKP describes the communication parameter list between SNS and the privilege policy module.

SNPEVS

Create subscription interface.This interface describes the parameter list allowing the creation of a temporary subscription from any BS2000 product or application. The call to SNS must be done by the SNPREQT interface.

SNPREQT

Access to the Notification Service. This interface is the only one allowing the call to SNS. All other API's are only parameter lists that must be anchored in the SNPREQT interface.

SNPREV

Raise Event interface. This interface describes the parameter list allowing the raise of an event from any BS2000 product or application. The call to SNS must be done by the SNPREQT interface.

SNRKERN

TU subsystem of SNS.

SNRTP

TPR subsystem of SNS.

SNS

SNS (SPOOL Notification Service) is a BS2000 subsystem that provides a tool for sending and managing notifications in the frame of BS2000.

SNS subsystem

The SNS subsystem with TU and TPR parts integrates the subsystem management, secured access to the notification resources file and generation of notifications:

– Notification Service TPR kernel (SNRTP)
– Notification Service TU kernel (SNRKERN).

subscriber

User who defines a subscription.

subscription

Notification resource. A subscription is a request done by a user to SNS in order to be notified when some events occurred.
When an event occurs, the subscription specifies to SNS how to send event notifications, where to send them and what to put into them.
Two types of subscriptions exist:

– permanent subscriptions addressing all the occurrences of an object class. Permanent subscriptions are registered with the Notification Resources Manager and are stored in the notification resources file NOTIFICATION.PARAMETERS.

– temporary subscriptions addressing a single occurrence of an object class. This occurrence is identified by an object ID whose format completely depends on the product or application raising the events.Temporary subscriptions are registered with a specific interface if such an interface is provided by the product or application raising the events.
A temporary subscription can be automatically removed at the end of the life of the corresponding object instance provided that a terminal event occurred. It is necessary that at least one event relative to the object class has been defined with the property TYPE=*TERMINAL.

template

File that can be used to customise the text (body part) of a notification delivery method.

Related publications

The Fujitsu Siemens Computers GmbH manuals listed below are available from these sources:

- On the Internet from our manual server (URL: *http://manuals.fujitsu-siemens.com*)
- On the CD-ROM “BS2000/OSD SoftBooks”.
  This CD contains practically all the language specific manuals and readme files for the BS2000 system software of the latest BS2000/OSD version.

Printed copies of the manuals can be purchased separately over the Internet from: *http://FSC-manualshop.com*.

BS2000/OSD-BC
Commands (Volume 1 -5)
User Guides

Target group
The manual addresses both nonprivileged BS2000/OSD users and systems support.
Contents
This manual contains BS2000/OSD commands (basic configuration and selected products) with the functionality for all privileges. The introduction provides information on cmd input.

BS2000/OSD-BC
Commands, Volume 6, Output in S Variables and SDF-P-BASYS
User Guide

Target group
This manual is addressed to programmers and users who write procedures.
Contents
Volume 6 contains tables of all S variables that are supplied with values by the SHOW commands in conjunction with structured output. Further chapters deal with:
– introduction to working with S variables
– SDF-P-BASYS

BS2000/OSD-BC
Executive Macros
User Guide

Target group
The manual addresses all BS2000/OSD assembly language programmers.
Contents
The manual contains a summary of all Executive macros, detailed descriptions of each macro with notes and examples, including job variable macros, and a comprehensive general training section.

Distributed Print Services (BS2000/OSD)
Printing in Computer Networks
User Guide

Target group
This manual is intended for nonprivileged users, device administrators and systems support of BS2000/OSD.
Contents
The manual provides descriptions of the principles, use and administration of Distributed Print Services for each of these user groups. Possible uses of Distributed Print Services are illustrated by examples.

DSSM/SSCM
Subsystem Management in BS2000/OSD
User Guide

Target group
This manual addresses systems support staff and software consultants of BS2000/OSD.
Contents
The following are described: BS2000/OSD subsystem concept, dynamic subsystem management (DSSM), subsystem catalog management (SSCM) and the associated commands and statements.

IMON (BS2000/OSD)
Installation Monitor
User Guide

Target group
This manual is intended for systems support staff of the BS2000/OSD operating system.
Contents
The manual describes the installation and administration of BS2000 software using the IMON installation monitor and its three components IMON-BAS, IMON-GPN and IMON-SIC. Installation (standard and customer-specific) using the component IMON-BAS for systems as of BS2000-OSD V3.0 is described in detail.

***inter*Net Value Edition** (BS2000/OSD)
User Guide

Target group
This manual is intended for network planners, generators and administrators who wish to use Mail Service in BS2000/OSD.
Contents
*inter*Net Value Edition is a supplement to *inter*Net Services that is available free of charge. The manual introduces the components of *inter*Net Value Edition and provides information on the installation, administration and operation of Mail Service in BS2000/OSD.

JV (BS2000/OSD)
Job Variables
User Guide

Target group
The manual addresses both nonprivileged users and systems support.
Contents
The manual describes management and possible uses of job variables. The command descriptions are divided according to function areas. The macro calls are described in a separate chapter.

OPG MAIL für BS2000
Benutzerhandbuch
(available in the internet, URL: *http://www.opg.de/produkte/mail.html*) in GERMAN ONLY

Target group
The manual addresses both nonprivileged users and systems support.
Contents
With the help of the MAIL program you can distribute emails from the BS2000/OSD to any internet/intranet user. The focus of application is sending emails on BS2000 procedures. For example you can mail fatal errors in time-critical batch processes to all people, who have to be informed immediately or distribute print data via mail right away instead of printing them and forward them via post. With the help of the MAILR programme you can receive emails in the BS2000/OSD from a mail server reachable within the network.

SDF (BS2000/OSD)
Introductory Guide to the SDF Dialog Interface
User Guide

Target group
BS2000/OSD users
Contents
This manual describes the interactive input of commands and statements in SDF format. A Getting Started chapter with easy-to-understand examples and further comprehensive examples facilitates use of SDF. SDF syntax files are discussed.

SECOS (BS2000/OSD
Security Control System
User Guide

Target group
– BS2000 system administrators
– BS2000 users working with extended access protection for files

Contents
Capabilities and application of the functional units:
– SRPM (System Resources and Privileges Management)
– SRPMSSO (Single Sign On)
– GUARDS (Generally Usable Access Control Administration System)
– GUARDDEF (Default Protection)
– GUARDCOO (Co-owner Protection)
– SAT (Security Audit Trail).

SPOOL (BS2000/OSD)
User Guide

Target group
This manual is intended for nonprivileged users, Spool & Print administrators, RSO device administrators and systems support staff.

Contents
The manual describes the operation of SPOOL.

Spool & Print - Commands (BS2000/OSD)
User Guide

Target group
This manual is intended for nonprivileged users, device administrators, cluster administrators, SPOOL administrators and system support staff.

Contents
The commands available for SPOOL, Dprint, RSO, SCSIPCL and SPS are described, but not those for subsystem management and job control.

Spool & Print - Macros and Exits (BS2000/OSD)
User Guide

Target group
The manual is intended for programmers who wish to address the Spool & Print Services in their programs directly.

Contents
The manual describes the macros and exits of the Spool & Print Services, including the macros for virtual printers. The description of the macros is arranged according to functions.

SPSERVE (BS2000/OSD)
User Guide

Target group
This manual is addressed to nonprivileged users, RSO device administrators, Dprint cluster administrators and those responsible for BS2000/OSD system operation.
Contents
The manual describes the SPSERVE utility routine with all its statements. It takes account of all extensions to SPOOL, RSO, SPCONV, PRM, Distributed Print Services, and SPS.

Wprint (Windows)

Target group
Users who want to print from within Windows applications and system administrators of SINIX, UNIX and BS2000/OSD systems as well as SINIX Spool administrators.
Contents
This manual describes the operation and functions of the Wprint-Server and Wprint-Client components as well as their installation and configuration.

Index

Contents

SNS V1.0B (BS2000/OSD)
SPOOL Notification Service

User Guide

Target group
The manual addresses nonprivileged users and systems support of BS2000/OSD.
Contents
This manual describes the SNS subsystem, which provides a tool for sending and managing notifications in the frame of BS2000/OSD.

Edition: October 2004

File: sns.pdf

This manual was produced by
cognitas. Gesellschaft für Technik-Dokumentation mbH
www.cognitas.de

Fujitsu Siemens computers GmbH
User Documentation
81730 Munich
Germany

Fax: (++49) 700 / 372 00000

e-mail: manuals@fujitsu-siemens.com
http://manuals.fujitsu-siemens.com

Comments
Suggestions
Corrections

Submitted by

Comments on SNS V1.0B
SPOOL Notification Service

Information on this document

On April 1, 2009, Fujitsu became the sole owner of Fujitsu Siemens Computers. This new subsidiary of Fujitsu has been renamed Fujitsu Technology Solutions.

This document from the document archive refers to a product version which was released a considerable time ago or which is no longer marketed.

Please note that all company references and copyrights in this document have been legally transferred to Fujitsu Technology Solutions.

Contact and support addresses will now be offered by Fujitsu Technology Solutions and have the format *...@ts.fujitsu.com*.

The Internet pages of Fujitsu Technology Solutions are available at *http://ts.fujitsu.com/*...
and the user documentation at *http://manuals.ts.fujitsu.com*.

Hinweise zum vorliegenden Dokument

Zum 1. April 2009 ist Fujitsu Siemens Computers in den alleinigen Besitz von Fujitsu übergegangen. Diese neue Tochtergesellschaft von Fujitsu trägt seitdem den Namen Fujitsu Technology Solutions.

Das vorliegende Dokument aus dem Dokumentenarchiv bezieht sich auf eine bereits vor längerer Zeit freigegebene oder nicht mehr im Vertrieb befindliche Produktversion.

Bitte beachten Sie, dass alle Firmenbezüge und Copyrights im vorliegenden Dokument rechtlich auf Fujitsu Technology Solutions übergegangen sind.

Kontakt- und Supportadressen werden nun von Fujitsu Technology Solutions angeboten und haben die Form *...@ts.fujitsu.com*.

Die Internetseiten von Fujitsu Technology Solutions finden Sie unter *http://de.ts.fujitsu.com/*..., und unter *http://manuals.ts.fujitsu.com* finden Sie die Benutzerdokumentation.