

rvsX

Version 3.05

User Manual

The products listed in this manual are protected by copyright.

rvsX

Version 3.05

User Manual

© 2005 by gedas deutschland GmbH

Pascalstraße 11

10587 Berlin

This manual is protected by copyright. All rights reserved. No part of this book may be used or reproduced in any form or by any means including photocopies, microfilm or any other means or stored in a database or retrieval system without obtaining prior permission from gedas. Rights are also reserved as far as lectures, radio and television is concerned.

We reserve the right to make changes to the content of this manual without giving prior notice. gedas is not liable for technical or printing errors or defects in this manual. Moreover, gedas shall not be liable for damage which is directly or indirectly caused by delivery, performance and use of this material.

Contents

Contents	3
Change History	7
1 Introduction	8
1.1 Short Description of the System.....	8
1.2 Security and Offline Compression.....	11
1.3 Target Group	13
1.4 Structure of the rvs [®] Documentation.....	13
1.5 Representation means	14
2 Installation of rvsX	16
1.1 Prerequisites	16
2.1 Function keys	16
2.2 New Installation	17
2.3 Update Installation.....	22
2.4 License key	27
3 Configuration of rvsX	30
3.1 Customizing Station Table and Related Tables.....	30
3.1.1 Virtual Stations	32
3.1.2 Identification of rvs [®] Nodes	33
3.1.3 Station Table ST	34
3.1.4 Routing Table RT	34
3.1.5 Neighbour Node (NachbarKnoten) NK	35
3.1.6 ODETTE Parameters OP	36
3.1.7 How to release or delete an EERP in the HOLD or HOLD_IMMED status?	40
3.1.8 LU 6.2 parameters LU (only AIX, HP-UX and SINIX)	41
3.1.9 X.25 native Communications XP	44
3.1.10 TCP/IP parameters TC	48
3.1.11 Special Logic.....	49
3.2 Choosing Privileges for rvsX.....	49
3.2.1 Running rvs [®] high-privileged.....	49
3.2.2 Running rvs [®] low-privileged	50
3.3 Defining the TCP/IP Connection	50
3.3.1 rvsX Configuration for TCP/IP.....	50
3.3.2 Problem Solving for TCP/IP	52
3.4 Defining the ISDN Network for BRICK Router	53
3.4.1 BRICK Router Installation	53
3.4.2 rvsX Configuration for BRICK Router	54
3.4.3 Problem Diagnosis for BRICK Router.....	56
3.5 Defining the ISDN Network for netISDN Base Software (only AIX Systems)	57
3.5.1 Adapter Installation for AIX	57
3.5.2 rvsX Configuration for AIX	57
3.5.3 Diagnosis for AIX	58
3.6 Defining the ISDN Network for internal ISDN board (only SINIX Systems).....	60
3.6.1 Configuring TNS for SINIX.....	60
3.6.2 Configuring rvsX (SINIX) rdstat.dat for ISDN communication.....	60
3.7 Defining the X.25 Network for AIX	61
3.7.1 Adapter Installation for AIX	61
3.7.2 rvsX Configuration for AIX	61
3.7.3 X.25 Problem Diagnosis for AIX	62
3.7.4 X.25 Routing Information for AIX	62
3.8 Defining the X.25 Network for SINIX.....	63
3.8.1 rvsX Configuration for SINIX.....	63
3.8.2 Tracing X.25 packet level for SINIX.....	64
3.9 Defining the X.25 Network for HP-UX	64

3.9.1	Configuration of HP-UX for X.25	64
3.9.2	rvsX Configuration for HP-UX	65
3.10	Defining the SNA Network for HP-UX	67
3.11	Defining the SNA Network for AIX and SINIX	75
3.11.1	Host Definition for AIX and SINIX	75
3.11.2	SNA Server/6000 Definition for AIX	80
3.11.3	SINIX TRANSIT Definition	87
3.12	Specify System Environment.....	87
3.13	Defining your Error Handling	90
4	rvsX Monitor.....	91
4.1	Starting rvsX Monitor (rvsstart).....	91
4.2	Stopping rvsX Monitor (rvsstop)	92
4.3	Stopping MasterTransmitter	93
4.4	Killing rvs [®] Programs (rvskill).....	94
4.5	Using non default Database	94
4.6	Setting rvs [®] Parameters at Start Time	95
4.7	Monitor Initial File rdmini.dat.....	95
4.8	Command Line Arguments.....	96
4.9	Return Codes	97
5	How to Work Interactively with rvs	99
5.1	Input and Output Fields	99
5.2	Function Keys.....	99
5.3	Dialog Interface (rvsdia).....	100
5.4	Interactive Sending and Receiving.....	103
5.4.1	Create Send Requests	103
5.4.2	Display Send Requests and Received Transmissions	110
5.4.3	Modify Send Requests and Received Transmissions	122
5.5	Interactive Administration	123
5.5.1	Resident Receive Entries.....	124
5.5.2	Job Start after Send Attempt.....	136
5.5.3	User List	143
5.5.4	Information about rvs [®]	148
6	Database Maintenance	149
6.1	Backup.....	149
6.2	Recovery	150
6.3	Dump rvs [®] Database (rvsddb)	151
6.4	Dump rvs [®] User and Job Starts (rvsdru)	151
6.5	Delete rvs [®] Database (rvsdbdel)	152
6.6	Create new Database (rvsidb)	152
6.7	Write rvs [®] Database (rvswdb).....	153
6.8	Cleanup	153
7	Operator Console and Commands.....	154
7.1	Work with Individual rvs [®] Commands	155
7.1.1	List information about one command.....	155
7.1.2	List Command Numbers	157
7.1.3	Hold, Free or Delete an rvs [®] command	157
7.2	Suspend Traffic to Neighbor.....	157
7.3	Activate Neighbor	158
7.4	Work with Stationtable Entries	158
7.5	Work With rvs [®] Parameters.....	159
7.6	Patterns	160
7.7	Command Descriptions	160
8	rvs[®] Parameters	164
8.1	rvs [®] Parameters' Overview	164
8.2	rvs [®] Parameter Values	175
8.2.1	Safety, Resource Consumption and Performance.....	175
8.2.2	Limit Number of Concurrent Senders	176

8.2.3	Limit Number of Concurrent X.25 or ISDN Receivers	177
8.2.4	TCP/IP Receiver	177
8.2.5	SNA Receiver.....	177
8.2.6	Optional Features.....	177
9	Configuration of Encryption: Key Administration.....	179
9.1.1	Creation of own Private and Public Key	180
9.1.2	Importing Keys (<i>rvskeyimp</i>).....	181
9.1.3	Support for public key certification.....	182
9.1.4	Distribution of Keys (<i>rvskeydst</i>).....	184
9.1.5	Deletion of Imported Keys (<i>rvskeydel</i>).....	184
9.1.6	Listing of Imported Keys (<i>rvskeylst</i>).....	185
9.2	Configuration of Offline Compression	186
10	Code Conversion.....	187
10.1	Automatic Code Conversion with <i>rvs</i> System Code Tables.....	188
10.2	Code Conversion with User Code Tables	189
10.2.1	Structure of the Code Conversion Tables.....	189
10.3	How to Carry out a Code Conversion.....	190
10.3.1	Code Conversion when Sending Files.....	190
10.3.2	Code Conversion when Receiving Files	192
11	<i>rvsX</i> Oracle Binding (<i>rvsX</i> High Performance)	194
11.1	Configuration of ORACLE	194
11.2	Configuration of <i>rvs</i> [®]	195
11.2.1	<i>rvs</i> [®] and Oracle Working on the Same Computer	195
11.2.2	<i>rvs</i> [®] and Oracle Working on Two Different Computers	196
12	<i>rvs</i>[®] Data Center	197
12.1	Introduction	197
12.2	System requirements.....	197
12.3	Installation.....	197
12.3.1	Installation of the first <i>rvs</i> [®] node	197
12.3.2	Adding an <i>rvs</i> [®] node	200
12.3.3	Updating all nodes (release change)	200
12.3.4	Updating all nodes (patch update).....	200
12.4	How to control fail safety	201
12.4.1	Monitor parameters.....	201
12.4.2	CNTMA and CNTGC	204
12.4.3	Service Provider parameter	204
12.5	How to start and stop <i>rvs</i> [®] Data Center.....	205
12.5.1	<i>rvs</i> [®] Data Center start	205
12.5.2	<i>rvs</i> [®] Data Center stop	206
12.5.3	<i>rvs</i> [®] node start.....	207
12.5.4	<i>rvs</i> [®] node stop.....	207
12.6	Working with <i>rvs</i> [®] Data Center	208
12.6.1	The batch interface (<i>rvsbat</i>).....	208
12.6.2	<i>rvs</i> [®] Client/Server.....	208
12.7	Low resources	209
12.7.1	Failure of central database/central directories.....	209
12.8	Logging	210
12.9	Parameter changes at runtime	213
	Glossary.....	215
	Index.....	220

Change History

The following changes of User Manual were made in the previous releases (including the current release):

Version 3.05:

New: Chapter 12 „rvs® Data Center“.

1 Introduction

In this chapter you will find a short description of rvs[®], its network architecture and the new features as well as an explanation of the structure of the rvs[®] documentation and of the target group.

1.1 Short Description of the System

What rvs[®] is

The abbreviation rvs[®] stands for the German word RechnerVerbundSystem. The rvs[®] computer communication system is a well established base service for electronic data interchange, EDI.

Task of the system is, to guarantee transmission of electronic data between heterogeneous computer platforms using different network protocols.

To do so, rvs[®] realizes a universal network model which can be configured by you within each network node.

It provides an efficient and reliable transport service for both standardized EDI message types and files of any format or content. You can get only data files which have been provided explicitly by rvs[®]. This means that rvs[®] does not provide an unauthorized access to remote or to own data files.

rvs[®] uses the OFTP protocol. The Volkswagen AG has been developed an extension to the standard OFTP: SNA LU6.2.

This "portable" version of rvs[®] has been developed in order to complement the classical rvs[®] product line which is based on MVS and VSE mainframes for use on midrange, mini systems and personal computing. Though the design and make of this product line differs considerably from the mainframe versions, the functional spectrum is almost exactly the same as that of rvs[®] MVS.

What rvs[®] is not

rvs[®] is not an online system. It neither supports direct terminal-like access to other sites, nor does it provide a communication pipe from application to application on a data record level to the end user. You cannot directly execute transfers in your own application, you rather can place send orders in the rvs[®] database which will be handled asynchronously.

rvs[®] is not a job scheduling system.

rvs[®] does not care about the contents of the data sets it is transporting. It only acts as a transparent transport medium and performs no semantical interpretation of the data it carries.

rvs[®] is not a EDI converter. But additional components for converting between specific message formats (e.g. VDA, ODETTE, EDIFACT, XML) can be purchased via gedas deutschland GmbH.

rvs[®] is not a network control or monitoring tool.

Supported Platforms

The following versions of portable rvs[®] are currently available:

- rvsX for SINIX using LU 6.2, X.25, ISDN and TCP/IP
- rvsX for AIX using LU 6.2, X.25, ISDN and TCP/IP
- rvsX for HP-UX using LU 6.2, X.25, ISDN and TCP/IP
- rvsX for IRIX using ISDN and TCP/IP
- rvsX for Linux using ISDN and TCP/IP
- rvsX (Linux/zSeries) using ISDN and TCP/IP
- rvsX for Solaris using ISDN and TCP/IP
- rvs400 for IBM OS/400 systems using LU 6.2, X.25, ISDN and TCP/IP
- rvsNT for MS Windows NT/2000 using LU 6.2, X.25, ISDN and TCP/IP
- rvsXP für MS Windows 2000 / XP and WS 2003 using X.25, ISDN and TCP/IP

Basic Functional Characteristics

The main function of rvs[®] is the reliable transfer of files. rvs[®] is suitable both for the transfer of large files and for the transfer of many small files. rvs[®] supports many networks with many stations which are based on different platforms. Various data formats are supported thereby.

The key characteristics of portable rvs[®] are:

- The Monitor, the central component of rvs[®], controls as a kind of local agent all work to be done. rvs[®] works asynchronously, i.e. its processing is not under your direct control. You just places a send order in the rvs[®] database by means of a menu controlled user interface or out of a batch file or program. The rvs[®] processes the send orders as soon as possible. The advantage is, that you or your application program does not need to wait for completion of a file transfer.
- The connection to the communication partner is automatically established when necessary.
- The automatic submission of jobs after reception of a data set is supported by means of resident receive entries. Wildcarding of data set names, user IDs, and station IDs is supported.
- The file transfer is possible for the following record formats:
 - U Unstructured,
 - T Text file,
 - F Fixed,

V Variable.

- Extensive security and authorization checking is performed.
- rvs[®] automatically repeats the transmission after an unsuccessful connection attempt or disruption of a connection. In the event of disruption rvs[®] only transmits the parts of a file which have not yet been transmitted. Therefore it is suited particularly to transmit large quantities of data even under difficult conditions.
- The compression of data during transfer increases transmission line throughput.
- The encryption enables the security of data during transmission.
- The user interface is a graphic interface.
- Besides a graphic interface a single command line interface is provided. That means that you can call rvs[®] within batch or command files.
- A program call interface allows you to call rvs[®] services out of a user application program.
- The ODETTE File Transfer Protocol, OFTP, is used.
- The ISDN, X.25, TCP/IP and SNA LU6.2 communication are supported.
- Message LOG files are provided for revision purposes.
- Tracing capabilities are provided on line I/O and/or OFTP level.
- rvs[®] supports multiple languages in messages, operator (console) and user interfaces (presently English and German).
- The data conversion, if wanted, can be carried out. Text files are automatically converted to the code (EBCDIC or ASCII) used on the target partner system.
- The earliest date/time of a transmission can be defined for each file to be transferred.
- A serialization facility allows the transfer of files in a sequence which you can define.

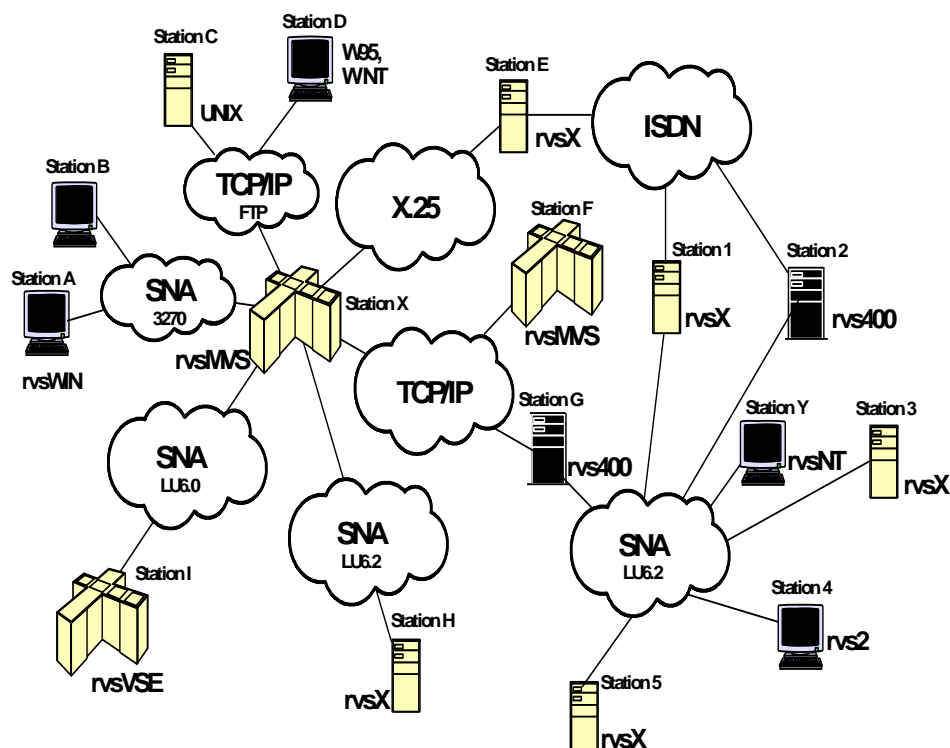
rvs[®] Network Architecture

A rvs[®] node (station) represents a 'peer' in a logical point-to-point connection. The connection is part of a logical network which consists of other rvs[®] installations.

rvs[®] always requires the presence of communication partners 'talking' the same high-level File Transfer Protocol. Therefore the partner must be another rvs[®] system or a compatible one.

In a logical network of rvs[®] nodes, the physical network can be quite inhomogeneous. That means that the kind of physical connection between two neighboring rvs[®] nodes is of bilateral interest only.

rvs[®] thus supports data transfer through an inhomogeneous physical network.



Picture 1 A rvs[®] Station Net Example with Different Platforms and Network Access Methods

This picture shows a logical network of rvs[®] stations. The nodes of this network are different computer platforms connected by different network access methods. Any number of stations can be connected in such a network.

This example shows the following relationships between the partners:

The rvsMVS station X is connected with a rvsX station H via LU6.2, but with a rvs400 station G via TCP/IP. Another rvsMVS station (station F) is also connected via TCP/IP. Other partners can be reached via SNA or X.25.

rvs[®] offers store and forward file transfer using gateway functionality, i.e. station H sends data to station G via Station X, even though there is no direct connection between the two stations. In doing so, rvs[®] routes the file between different network protocols (to station X via SNA LU6.2 and to station G via TCP/IP) to its final destination. In the same way the station X could send data to the station 2.

1.2 Security and Offline Compression

Beginning with rvs[®] release 2.05 a set of new powerful features is introduced that usefully extends the capabilities of the rvs[®] file transfer solution:

- rvs[®] now is able to encrypt and decrypt files in order to ensure privacy of the data when being transmitted over a network especially useful for none-private wide area networks, e.g. when using the internet.
- data may be compressed offline before network connection is established, in this way network costs are significantly reduced.

These new features can be used when both partner stations (sender and receiver) use rvs[®]. OFTP routing is not affected. Later a stand alone offline encryption and compression product will be available that could be used in conjunction with other OFTP products than rvs[®].

The Odette protocol VDA 4914/2 is extended to enable transmission of encrypted and compressed files: the first data block is extended by a rvs[®] managed header.

All formats and character sets are supported.

Both – encryption and offline compression – work automatically after have been configured once on a station by station base. The new features work directly between sender and receive station by tunneling routing stations. From there compression and encryption are available, if the sender and receive station use rvs[®]. Routing stations on the way don't have to assist the extended features.

On MVS systems the new encryption and compression subsystem enables extended format information which prevents loss of file format information (e.g. blocksize, blocked, ASA).

Encryption Features

The security of rvs[®] uses both a symmetric cipher (3DES) and a public/private-key cipher (RSA). A symmetric cipher is a cipher that uses the same key for both encryption and decryption, the public/private-key manner uses two keys for encryption and decryption.

The symmetric cipher is stronger from a security standpoint, and the public/private-key encryption and decryption are more expensive than the corresponding operations in symmetric systems. The primary problem with symmetric ciphers is not their security but with key exchange and the number of the keys. With a private and public key pair it's possible to distribute the public key by using insecure communication channels and use one key with different partner stations.

The encryption works by using a public/private-key cipher to share a key for the symmetric cipher. The actual file being sent is then encrypted using the session key and sent to the recipient. Because the symmetric key sharing is secure, the symmetric key used is different for each file transmission. It is possible to create and use private and public keys with a size between 768 and 2048 bit.

Furthermore rvs[®] provides a document's digital signature, which protects the transmitted file against changes and check the ownership of the transmitted data. The implementation of this feature uses a hash function (SHA1). A hash function is a many-to-one function that maps its input to a value in a finite set. This value is encrypted with the private key of the sender to steady the identity of the file owner.

Offline Compression Features

The compression is done by using the GNU zip algorithms¹. rvs[®] uses a gzip compatible compression (rfc. 1952). The compression method reduces the data size in the best case down to a level of 30 % and essentially never expands the data. Only in the case of very small files the added process information will expand the absolute file size.

The compression is independent of CPU type, operating system and character set and provides a integrity check of the uncompressed data.

1.3 Target Group

This manual is intended as a reference for persons who are assigned to install and configure rvs[®] running in the day-to-day business.

rvs[®] is basically designed to run without operator control. However, there are always some duties left requiring hand-work. There is for example the very important task of maintaining the database. This requires regular cleanups and backups.

The following skills are required to be able to use rvs[®]:

- good knowledge of the current operating system
- knowledge of the communications technics in use
 - SNA LU 6.2 PU 2.1
 - X.25 native communications and/or ISDN native communications
 - TCP/IP

Before starting to work it is advisable to have read this book.

1.4 Structure of the rvs[®] Documentation

The rvs[®] Documentation consists of the following manuals:

- User Manual
The User Manual contains all important steps about installing and configuring rvsX. It shows you how to transfer files and handle the rvsX database. The new

¹ 1995-1998 Jean-loup Gailly and Mark Adler; for more information see Glossary

features such as encryption, off-line compression, and code conversion are described in this manual, too.

- Reference Manual
The Reference Manual is the common manual for rvs[®] portable (rvsX, rvsNT, rvs400). It contains descriptions of the C-Cal Interface (*rvscal*), Command Line Interface (*rvsbat*) and of the rvs[®] utilities. Furthermore, this manual contains information on the technical background of rvs[®].
- Messages and Return Codes Manual
This manual describes all messages and error codes which could be displayed on the rvs[®] Monitor and in the log file.

The User Manual and the Reference Manual are available as printed and as electronic documents. The electronic document is available in Portable Document Format (PDF). The "Messages and Return Codes Manual" is only available as electronic document.

1.5 Representation means

This chapter contains the description of the indications which are used in this manual and the explanation of the expressions which are marked.

Indications

<code>courier</code>	commands, menu commands, file names, path names, programs, examples, scripts, qualifiers, data sets, fields, options, modes, window names, dialog boxes and statuses
BOLD and IN CAPITAL LETTERS	parameters, environment variables, variables
"quotation mark"	links to other manuals, sections and chapters, literature
bold	important, names of operating systems, proper names, buttons, function keys

Expressions

rvsX is the synonym of rvs[®] for **UNIX** systems.

rvsNT is the synonym of rvs[®] for **Windows NT** systems.

rvsXP is the synonym of rvs[®] for **Windows XP / 2000** systems

rvs400 is the synonym of rvs[®] for **AS/400** systems.

Directories

As user directories are located on different locations for the different operating systems we use the variable **\$RVSPATH** in this manual. Default values are:

- /home/rvs/ for **AIX, Solaris, IRIX, Linux** and **SCO**
- /users/rvs/ for **HP-UX**
- /defpath/rvs/ for **SINIX**
- c:\rvs for **Windows NT / XP / 2000**

Substitute the variable with your correct path.

Generally, the file names on **OS/400** systems are always written in capital letters.

2 Installation of rvsX

rvs[®] allows installation of a new rvs[®] version or update of an existing rvs[®] installation (rvs[®] version 2.0 or later). To install rvs[®] please follow the instructions for a new installation or an update respectively in the specified order.

1.1 Prerequisites

To install rvs[®] on

- **AIX**, you need a RS/6000 computer system running AIX 4.3 or AIX 5.2. Supported protocols are: TCP/IP, ISDN(BinTec), X.25 and SNA(LU6.2).
- **Solaris**, you need an SUN Sparc computer system running Solaris 8. rvs[®] on SUN systems supports the protocols ISDN (BinTec) and TCP/IP.
- **IRIX**, you need an SGI computer system running IRIX 6.5. rvsX (**IRIX**) supports the protocols ISDN(BinTec) and TCP/IP.
- **Reliant UNIX**, you need an RM computer system with Reliant Unix 5.45. Supported networks are: TCP/IP, ISDN(native), ISDN(BinTec), X.25 and SNA(LU6.2). For X.25 communication you need a telecommunication processor, CCP-WAN-X.25 and CMX. For SNA(LU6.2) you need a a telecommunication processor or TR controller, CMX and TRANSIT system.
- **HP-UX**, you need a HP/9000 computer system running HP-UX 11.00 or HP-UX 11.11. Supported protocols are: TCP/IP, ISDN(BinTec), X.25 and SNA(LU6.2). For LU6.2 Connections you must have installed SNAplus APPC 9.7 or above. For X.25 Connections you must have installed X25/9000.
- **Linux**, you need a PC system running Linux, that supports libraries `glibc 2.2` and `glibc 2.3`. rvs[®] on Linux systems supports the protocols ISDN(BinTec) and TCP/IP.

ISDN(BinTec) means the communication with BinTec ISDN router containing remote CAPI interface (2-30 channels). The following ISDN routers were tested: X4100, BIANCA/BRIC-XS, -XM and -XL.

Note: The rvs[®] data sheets contain the actual version numbers of the supported operating system.

By default, rvs[®] is distributed on CD ROM or data tape, so your system must be able to read those. Please contact your distributor, if you have different requirements.

2.1 Function keys

In order to work with the rvs[®] Dialog Interface you need a keyboard with function keys <F1> - <F8>. Sometimes the keys <F1> - <F5> have a special meaning, so you must use the keys <PF1> - <PF4> instead of <F1> - <F4> and <F6> - <F9>

instead of <F5> - <F8>. On "vt100" terminal emulations, you can use the numeric keypad keys <1> - <9>.

If no function keys are available: The key

"?" works as <F1> (help),
"&" works as <F2> (add),
"!" works as <F3> (exit),
"%" works as <F5> (delete/refresh),
"<" works as <F7> (up),
">" works as <F8> (down).

2.2 New Installation

Before you may install rvsX it is necessary to define the user **rvs** and the home directory **\$RVSPATH** (see chapter 1.5 "Representation means" for the detailed description of **\$RVSPATH**). In this chapter is described the new installation to rvsX version 2.06. The new installation for the version 3.05 is almost identical.

1. Log in as root and create an UNIX user **rvs** (recommended user name) with home directory **\$RVSPATH** (recommended directory); in our following example `/home/skk/rvs`.
2. Log in as user **rvs**.
3. Copy and uncompress the installation file from CD ROM or tape respectively to the rvs[®] user home directory (e.g. **\$RVSPATH**, please refer to the readme file for the name of the installation file).

Example (Uncompression):

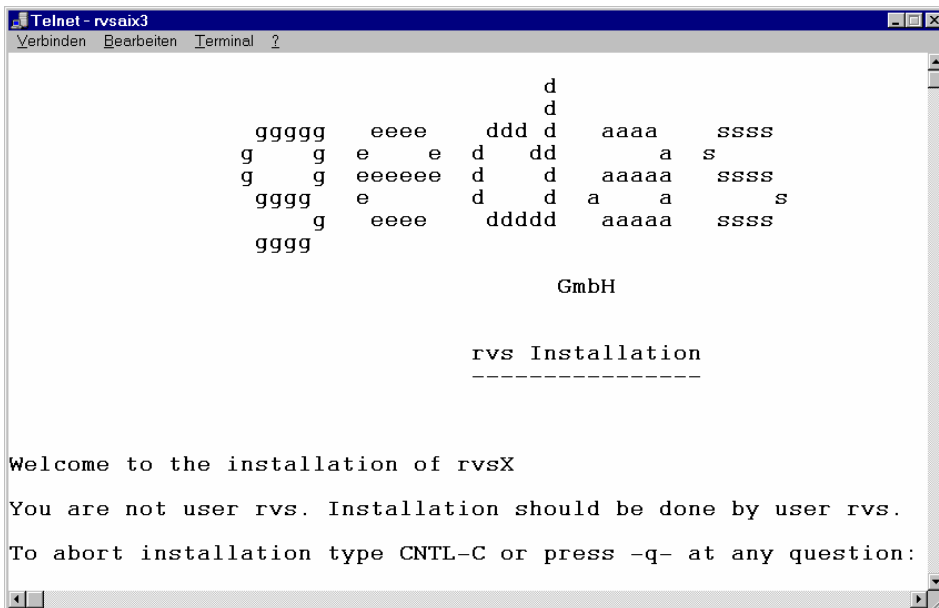
```
uncompress rvs_2_06_00.aix433.setup.Z
```

Result: The name of the uncompressed file is

```
rvs_2_06_00.aix433.setup.
```

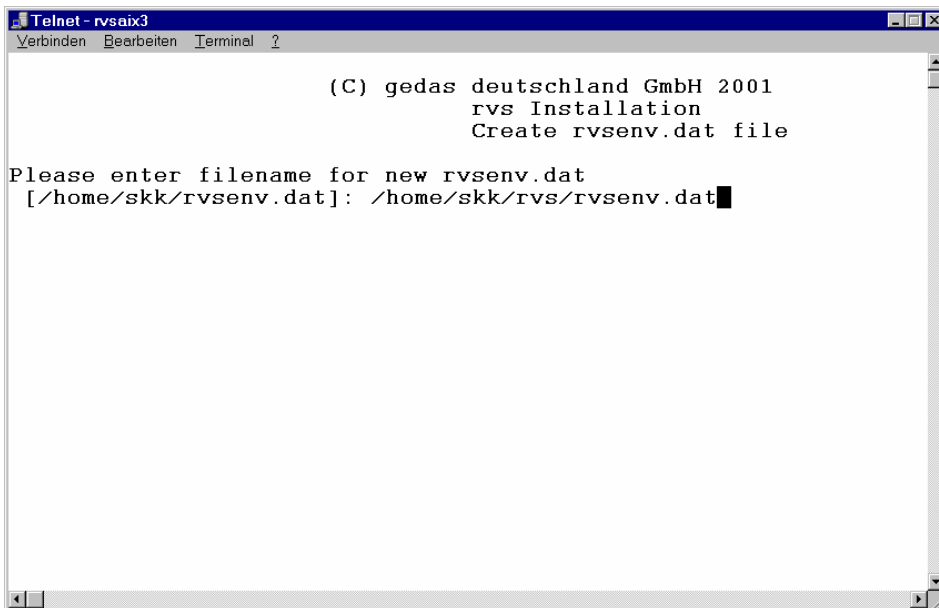
4. Run installation procedure by entering the name of the installation file (**Example:** `rvs_2_06_00.aix433.setup`) and pressing <Enter>.

5. The first screen welcomes you:

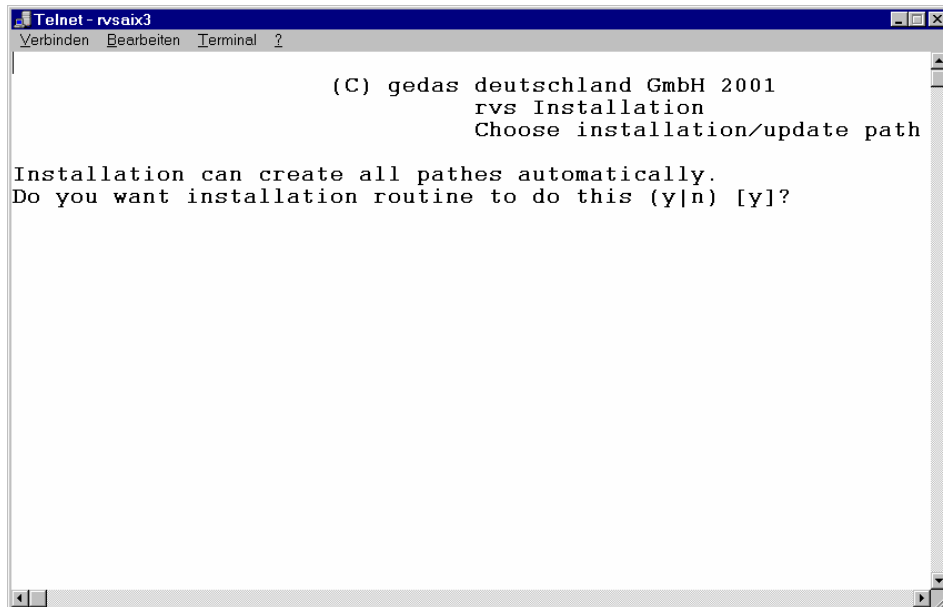


Press **<Enter>** to continue.

6. The next two screens assist you in configuring the rvs[®] environment. Enter path and name of rvs[®] environment file. The default value (\$RVSPATH/rvsenv.dat) is shown.



7. You may decide if you wish an automatic or custom selection of rvs[®] paths:



```

Telnet - rvsaix3
Verbinden Bearbeiten Terminal ?

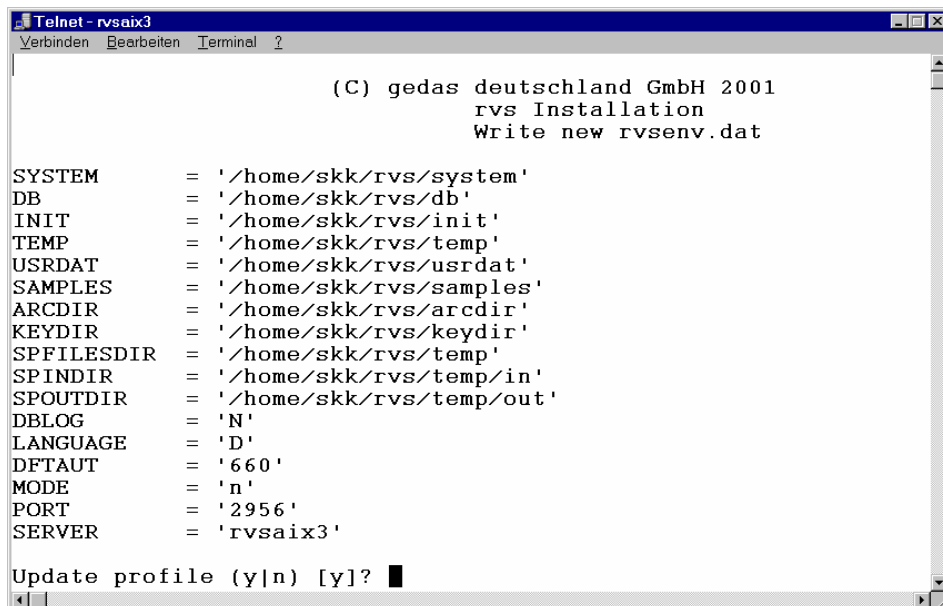
(C) gedas deutschland GmbH 2001
    rvs Installation
    Choose installation/update path

Installation can create all paths automatically.
Do you want installation routine to do this (y|n) [y]?

```

Automatically means, that all rvs[®] directories should be generated as subdirectories of **\$RVSPATH**

8. After you have entered the rvs[®] path definitions or have them created by the installation routine the next screen shows the results that will be written in the rvs[®] environment file:



```

Telnet - rvsaix3
Verbinden Bearbeiten Terminal ?

(C) gedas deutschland GmbH 2001
    rvs Installation
    Write new rvsendv.dat

SYSTEM      = '/home/skk/rvs/system'
DB          = '/home/skk/rvs/db'
INIT        = '/home/skk/rvs/init'
TEMP        = '/home/skk/rvs/temp'
USRDAT      = '/home/skk/rvs/usrdat'
SAMPLES     = '/home/skk/rvs/samples'
ARCDIR      = '/home/skk/rvs/arcdir'
KEYDIR      = '/home/skk/rvs/keydir'
SPFILESDIR = '/home/skk/rvs/temp'
SPINDIR     = '/home/skk/rvs/temp/in'
SPOUTDIR    = '/home/skk/rvs/temp/out'
DBLOG       = 'N'
LANGUAGE    = 'D'
DFTAUT      = '660'
MODE        = 'n'
PORT        = '2956'
SERVER      = 'rvsaix3'

Update profile (y|n) [y]? █

```

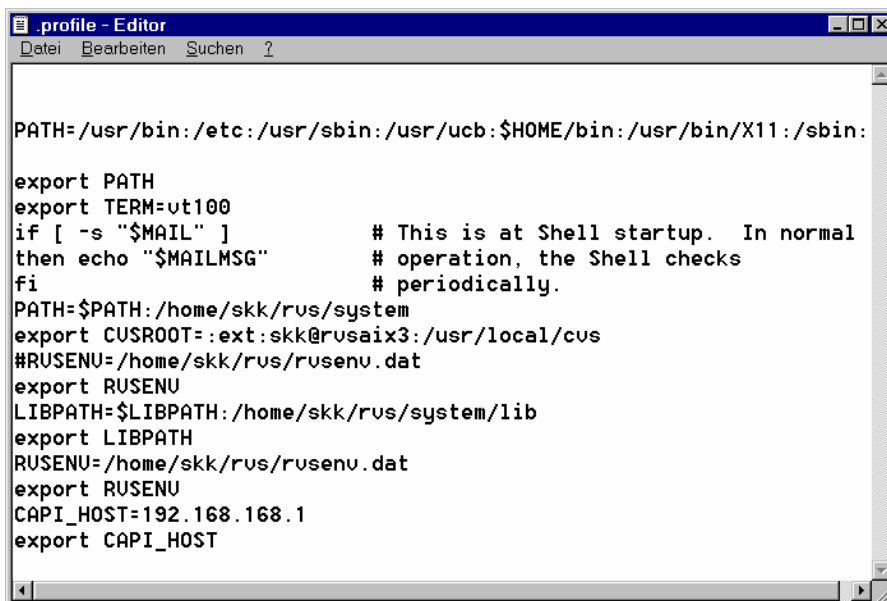
The installation routine may modify the configuration file of the Unix user (e.g. `$RVSPATH/.profile`) for you (press **y**). The variables that rvs[®] needs, will be added. These are: **RVSENV**, **PATH** and **LIBPATH** (**LD_LIBRARY_PATH** for LINUX Systems).

RVSENV: the name of the rvs[®] environment file.

PATH: this variable should be extended with `$RVSPATH/system`.

LIBPATH: this variable is needed for the rvs[®] "shared libraries". It should be extended with `$RVSPATH/system/lib`.

Example for a AIX configuration file (user `skk`):



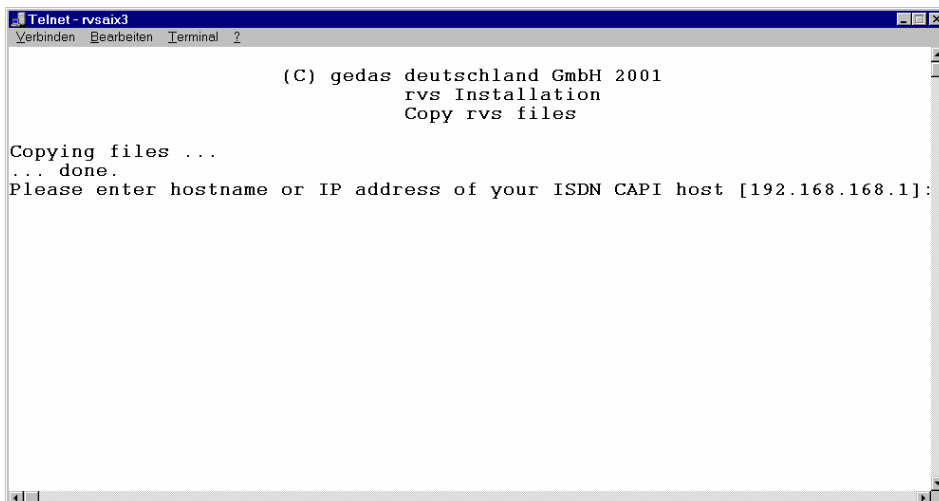
```
.profile - Editor
Datei Bearbeiten Suchen ?

PATH=/usr/bin:/etc:/usr/sbin:/usr/ucb:$HOME/bin:/usr/bin/X11:/sbin:

export PATH
export TERM=vt100
if [ -s "$MAIL" ]          # This is at Shell startup. In normal
then echo "$MAILMSG"      # operation, the Shell checks
fi                          # periodically.
PATH=$PATH:/home/skk/rvs/system
export CUSROOT=:ext:skk@rvisaix3:/usr/local/cvs
#RUSENU=/home/skk/rvs/rusenu.dat
export RUSENU
LIBPATH=$LIBPATH:/home/skk/rvs/system/lib
export LIBPATH
RUSENU=/home/skk/rvs/rusenu.dat
export RUSENU
CAPI_HOST=192.168.168.1
export CAPI_HOST
```

The variable **CAPI_HOST** identifies your ISDN router (if you use a BIANCA/BRICK ISDN router with the TCP/IP protocol). If you modify the UNIX configuration file by yourself, you should write here the IP address or the host name of the router.

9. The next screen informs about copying files and asks what is the IP address or the host name of your ISDN router.



```
Telnet - rvisaix3
Verbinden Bearbeiten Terminal ?

(C) gedas deutschland GmbH 2001
    rvs Installation
    Copy rvs files

Copying files ...
... done.
Please enter hostname or IP address of your ISDN CAPI host [192.168.168.1]:
```

If your ISDN router is not known to rvs[®], you should write here its IP address or its host name. In the example above, the IP address is already known and written in brackets (`[192.168.168.1]`).

10. This screen assists you in configuring your rvs[®] installation, in configuring your local station in the rvs[®] station table and in initializing the rvs[®] database:

```

(C) gedas deutschland GmbH 2001
    rvs Installation
    Configuration

Do you want to start rvs every times your system starts (y|n) [n]?
To break station configuration press -qq- instead of parameter value.
Please enter new SID for station LOC [LOC]:
Please enter value for parameter ST-STATNAME ['local rvs station']:
Please enter value for parameter NK-PROTOCOL [T]:
Please enter value for parameter OP-ODETTEID ['aaa']:
Please enter value for parameter TC-PORT [3305]: |
Please enter value for parameter XP-N [1]: |
Please enter value for parameter XP-LINK ["RCAPI1"]: |
Please enter value for parameter XP-ISDNNO [""]:
Do you want to change parameters in rdmini.dat (y|n) [n]? n
Create database now (y|n) [y]? y
Creating database ...
... done.
Continue: |
  
```

You can already write here the parameters, which are important for your local station (e.g. **OP-ODETTEID**, **TC-PORT** or **XP-ISDNNO**).

11. The last screen finishes the installation and informs you about the next required steps in order to complete the rvs[®] configuration.

```

(C) gedas deutschland GmbH 2001
    rvs Installation
    End

now continue with the following steps :
- see details in the INSTALLATION MANUAL !!-

1. check your licence key
2. customize the station table
3. start the monitor

Your profile was changed.
Please log out now and log in again so changes can take effect.

:~/home/skk> █
  
```

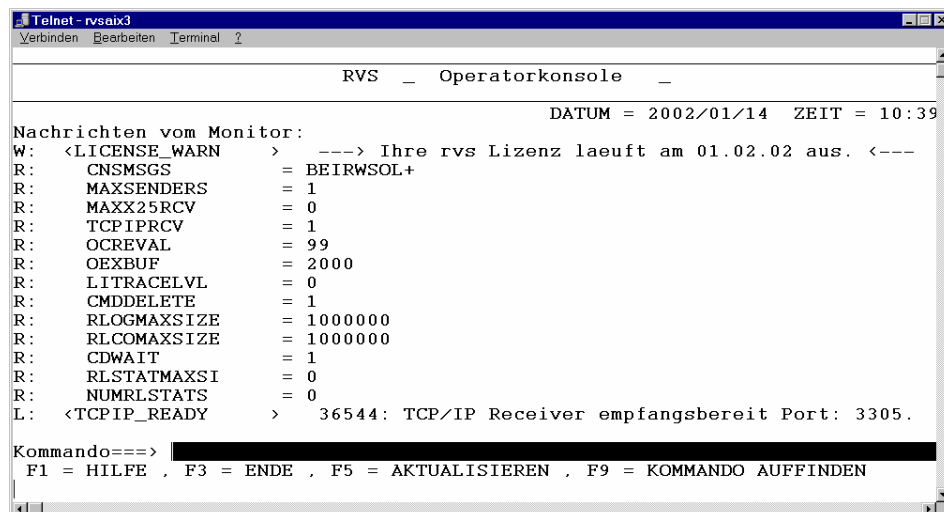
12. As the next step, you should log out and log in, so the changes you have made can take effect.

13. For test purpose, you should start and stop rvs[®] with the following command:

```
rvsstart
rvscns
```

The command `rvscns` starts the `rvs`[®] Operator Console, so you can follow the `rvs`[®] monitor messages.

A successful start appears as follows:



```
Telnet - rvsx3
Verbinden Bearbeiten Terminal ?

RVS _ Operatorkonsole _

Nachrichten vom Monitor:
                                DATUM = 2002/01/14  ZEIT = 10:39
W: <LICENSE_WARN    >  ----> Ihre rvs Lizenz laeuft am 01.02.02 aus. <----
R:  CNSMSGs         = BEIRWSOL+
R:  MAXSENDERS      = 1
R:  MAXX25RCV       = 0
R:  TCPIPRCV        = 1
R:  OCREVAL         = 99
R:  OEXBUF          = 2000
R:  LITRACELVL      = 0
R:  CMDDELETE       = 1
R:  RLOGMAXSIZE     = 1000000
R:  RLCOMAXSIZE     = 1000000
R:  CDWAIT          = 1
R:  RLSTATMAXSI     = 0
R:  NUMRLSTATS      = 0
L:  <TCPIP_READY    >  36544: TCP/IP Receiver empfangsbereit Port: 3305.

Kommando===>
F1 = HILFE , F3 = ENDE , F5 = AKTUALISIEREN , F9 = KOMMANDO AUFFINDEN
```

You can stop `rvs`[®] with the shell command: `rvsstop`.

14. Customize station tables (file `$RVSPATH/init/rdstat.dat`) as described in 3.1 "Customizing Station Table and Related Tables". The file `$RVSPATH/init/rdstat.dat` contains also some examples for the local and partner stations.
15. Modify the Monitor's initial commands in the file `$RVSPATH/init/rdmini.dat` to suit your environment (see chapter 4.7 "Monitor Initial File `rdmini.dat`"). The most important parameters in this file are: **TCPIPRCV**, **X25RCV** or **SNARCV**, depending on the fact what kind of network you use (TCPIP, X.25/ISDN or SNA). You are able to receive files only, if the value of the suitable parameter equals 1 (default) or more. For more information see chapter 8 "`rvs`[®] Parameters".
16. Choose and define privileges according to your requirements (see chapter 3.2 "Choosing Privileges for `rvsX`").

This completes the installation.

2.3 Update Installation

During the update installation the `rvs`[®] data will be saved and read again after the installation. But we recommend you to make a back up of the previous release, so if the installation fails, you can have your old data.

The following steps are to be done for a rvs[®] back up:

- Make a back up of the whole rvs[®] directory, especially of the license file (`rdkey.dat`) and a station table (`rdstat.dat`).
- Make a back up of a whole rvs[®] database (with a tool `rvsddb`, see chapter 6.3).
- Make a separate back up of users and job starts only (tool `rvsdru`, see chapter 6.4).

In this chapter is described the update to rvsX version 2.06. The update to the rvsX version 3.05 is almost identical.

1. Log in as user **rvs**. Change to rvs[®] user's home directory; in our following example `/home/skk/rvs`.
2. Copy and uncompress the installation file from CD ROM or tape respectively to the rvs[®] user home directory (e.g. **\$RVSPATH**, please refer to the readme file for the name of the installation file, see chapter 1.5 "Representation means" for the detailed description of **\$RVSPATH**).

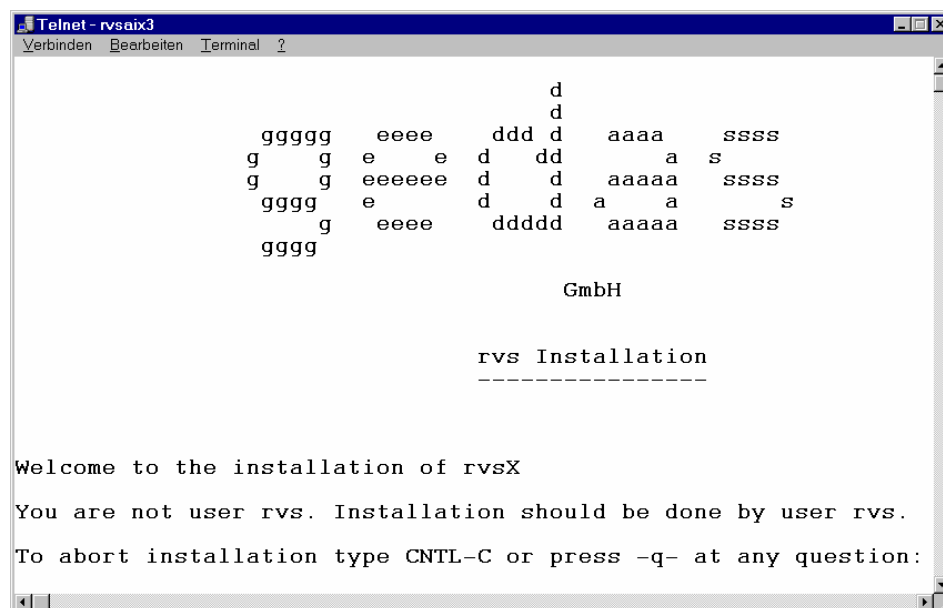
Example (Uncompression):

```
uncompress rvs_2_06_00.aix433.setup.Z
```

Result: The name of the uncompressed file is

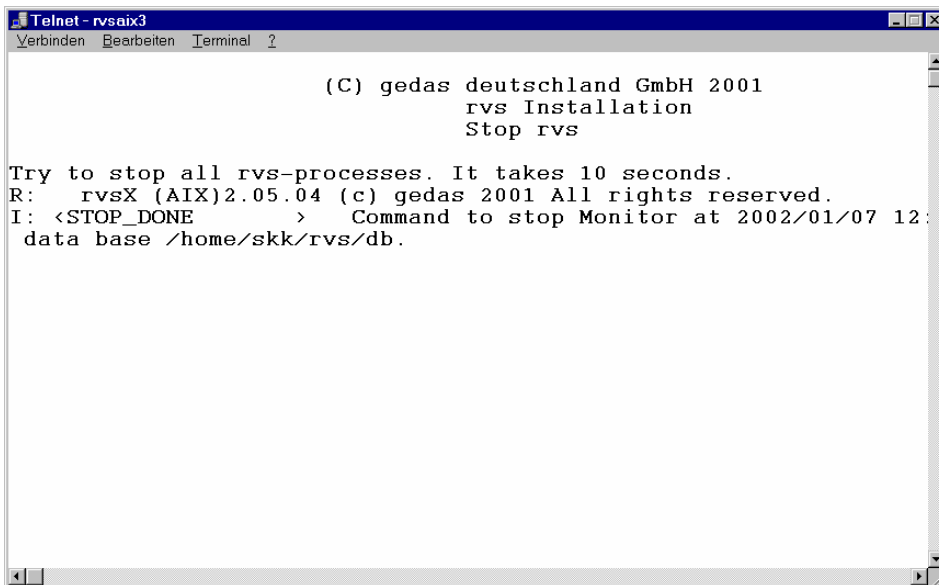
```
rvs_2_06_00.aix433.setup.
```

3. Run installation procedure by entering the name of the installation file (**Example:** `rvs_2_06_00.aix433.setup`) and pressing **<Enter>**.
4. The first screen welcomes you:



5. Press **<Enter>** to continue.

6. If rvs[®] is running, installation will stop all rvs[®] processes:

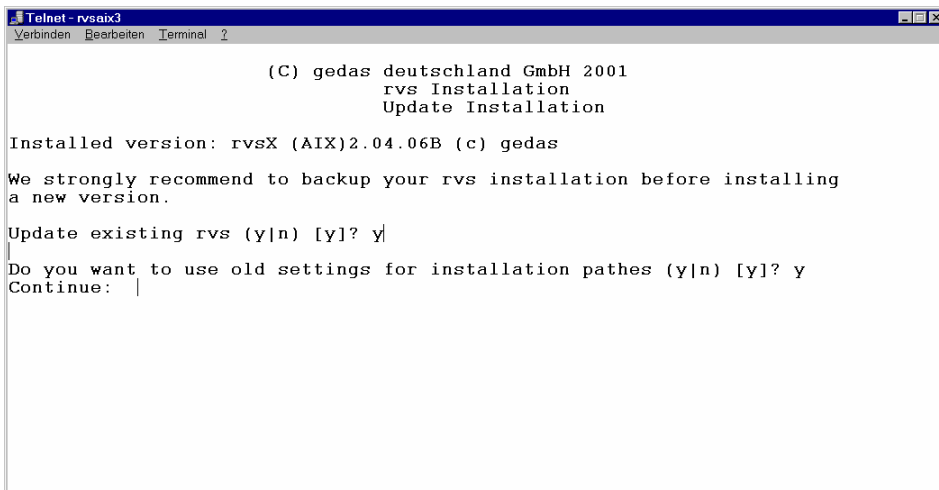


```
Telnet - rvsaix3
Verbinden Bearbeiten Terminal ?

(C) gedas deutschland GmbH 2001
    rvs Installation
    Stop rvs

Try to stop all rvs-processes. It takes 10 seconds.
R:   rvsX (AIX)2.05.04 (c) gedas 2001 All rights reserved.
I: <STOP_DONE > Command to stop Monitor at 2002/01/07 12:
    data base /home/skk/rvs/db.
```

7. The installation asks for updating existing rvs[®]. If you choose not to update, you will install rvs[®] like a new version (see also chapter "New Installation"). The following screens show an update.



```
Telnet - rvsaix3
Verbinden Bearbeiten Terminal ?

(C) gedas deutschland GmbH 2001
    rvs Installation
    Update Installation

Installed version: rvsX (AIX)2.04.06B (c) gedas

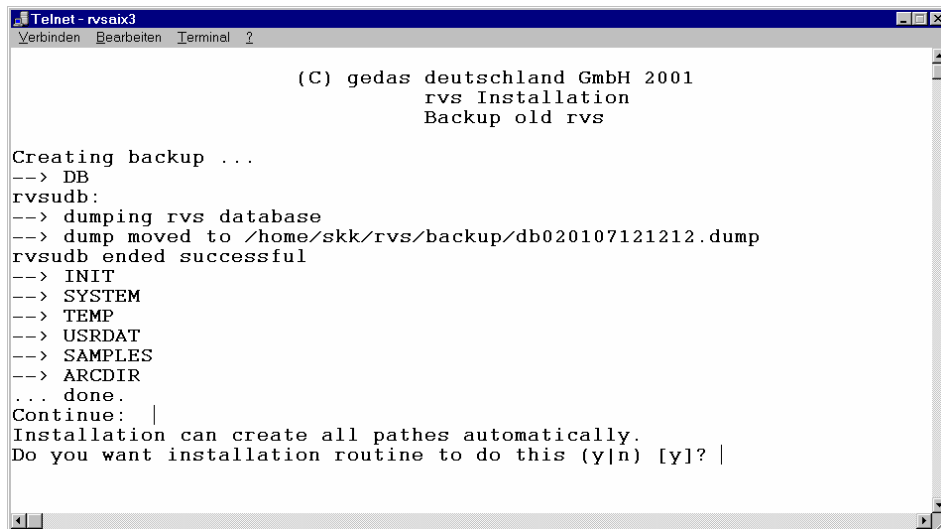
We strongly recommend to backup your rvs installation before installing
a new version.

Update existing rvs (y|n) [y]? y

Do you want to use old settings for installation pathes (y|n) [y]? y
Continue: |
```

"Old settings" are the values of the rvs[®] variables in the configuration file of the UNIX user (e.g. /home/skk/.profile) and in the rvs[®] environment file, see part 8 in chapter 2.2 "New Installation".

8. The Installation creates a backup and dumps the current database. It asks for automatically creation of all paths. Automatically means, that all rvs[®] directories will be generated as subdirectories of \$RVSPATH.

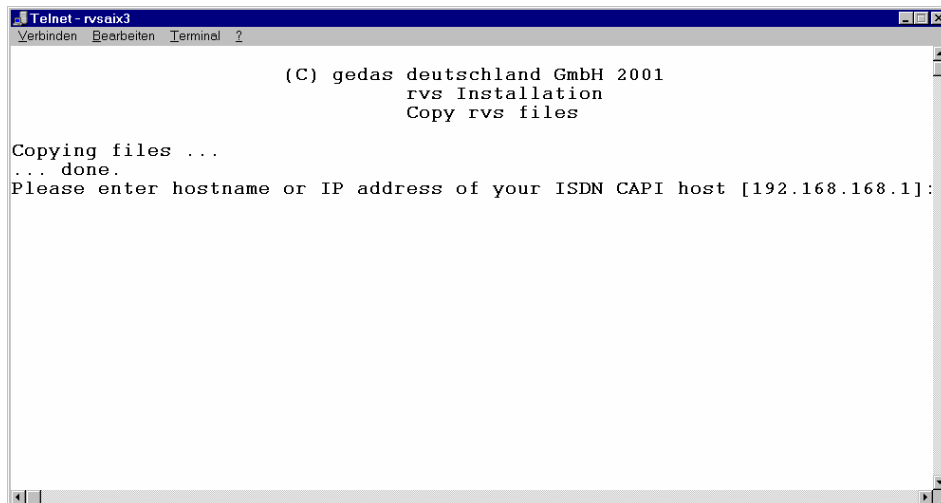


```
Telnet - rvsaix3
Verbinden Bearbeiten Terminal ?

(C) gedas deutschland GmbH 2001
    rvs Installation
    Backup old rvs

Creating backup ...
--> DB
rvsddb:
--> dumping rvs database
--> dump moved to /home/skk/rvs/backup/db020107121212.dump
rvsddb ended successful
--> INIT
--> SYSTEM
--> TEMP
--> USRDAT
--> SAMPLES
--> ARCDIR
... done.
Continue: |
Installation can create all paths automatically.
Do you want installation routine to do this (y|n) [y]? |
```

9. The next screen informs about copying files and asks what is the IP address or the host name of the your ISDN Router:



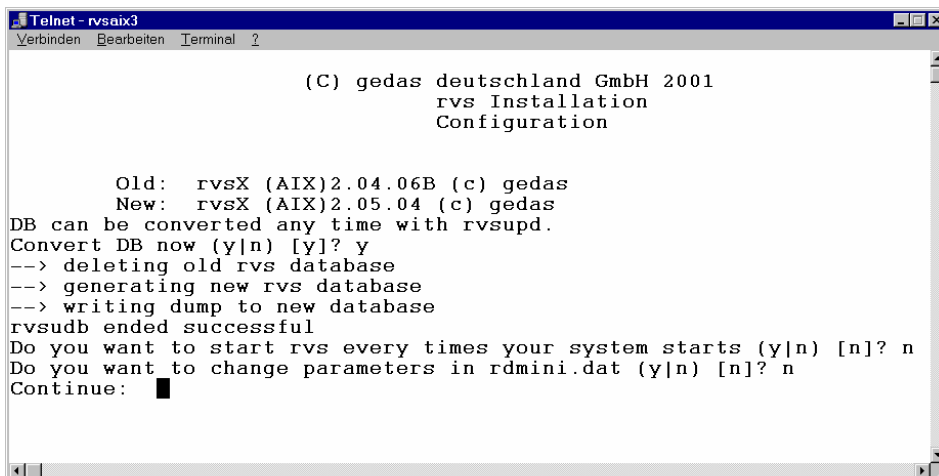
```
Telnet - rvsaix3
Verbinden Bearbeiten Terminal ?

(C) gedas deutschland GmbH 2001
    rvs Installation
    Copy rvs files

Copying files ...
... done.
Please enter hostname or IP address of your ISDN CAPI host [192.168.168.1]:
```

If your ISDN router is not known to rvs[®], you should write here its IP address or its host name. In the example above, the IP address is already known and written in brackets ([192.168.168.1]). See part 8 in chapter 2.2 "New Installation" for more information about **CAPI_HOST**.

10. The old database is deleted, a new one is created and the dump of rvs[®] database is written to the new database.

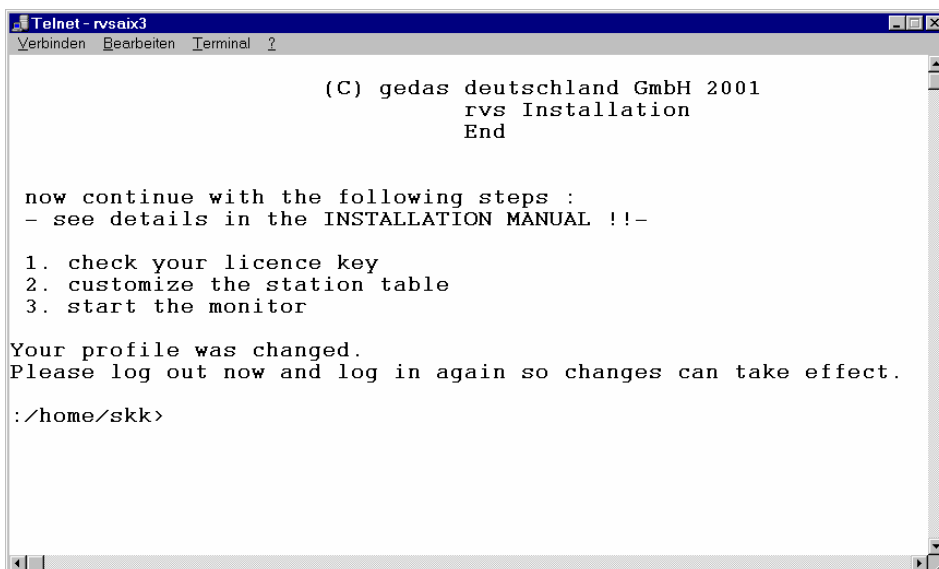


```
Telnet - rvsaix3
Verbinden Bearbeiten Terminal ?

(C) gedas deutschland GmbH 2001
    rvs Installation
    Configuration

Old: rvsX (AIX)2.04.06B (c) gedas
New: rvsX (AIX)2.05.04 (c) gedas
DB can be converted any time with rvsupd.
Convert DB now (y|n) [y]? y
--> deleting old rvs database
--> generating new rvs database
--> writing dump to new database
rvsddb ended successful
Do you want to start rvs every times your system starts (y|n) [n]? n
Do you want to change parameters in rdmini.dat (y|n) [n]? n
Continue: █
```

11. The last screen finishes the installation and informs you about the next required steps in order to complete the rvs[®] configuration.



```
Telnet - rvsaix3
Verbinden Bearbeiten Terminal ?

(C) gedas deutschland GmbH 2001
    rvs Installation
    End

now continue with the following steps :
- see details in the INSTALLATION MANUAL !!-

1. check your licence key
2. customize the station table
3. start the monitor

Your profile was changed.
Please log out now and log in again so changes can take effect.

:/home/skk>
```

12. As the next step, you should log out and log in, so the changes you have made can take effect.
13. For test purpose, you should start and stop rvs[®] with the following command:

```
rvsstart
rvscns
```

The command `rvscns` starts the rvs[®] Operator Console, so you can follow the rvs[®] monitor messages.

A successful start appears as follows:

```

Telnet - rvsaix3
Verbinden Bearbeiten Terminal ?

RVS _ Operatorkonsole _

DATUM = 2002/01/14 ZEIT = 10:39

Nachrichten vom Monitor:
W: <LICENSE_WARN > ----> Ihre rvs Lizenz laeuft am 01.02.02 aus. <----
R: CNSMSGs = BEIRWSOL+
R: MAXSENDERS = 1
R: MAXX25RCV = 0
R: TCPIPCV = 1
R: OCREVAL = 99
R: OEXBUF = 2000
R: LITRACELVL = 0
R: CMDDELETE = 1
R: RLOGMAXSIZE = 1000000
R: RLCOMAXSIZE = 1000000
R: CDWAIT = 1
R: RLSTATMAXSI = 0
R: NUMRLSTATS = 0
L: <TCPIP_READY > 36544: TCP/IP Receiver empfangsbereit Port: 3305.

Kommando===>
F1 = HILFE , F3 = ENDE , F5 = AKTUALISIEREN , F9 = KOMMANDO AUFFINDEN

```

You can stop rvs[®] with the shell command: `rvsstop`.

14. Customize station tables (file `$RVSPATH/init/rdstat.dat`) as described in 3.1 "Customizing Station Table and Related Tables". The file `$RVSPATH/init/rdstat.dat` contains also some examples for the lokal and partner stations.
15. Modify the Monitor's initial commands in the file `$RVSPATH/init/rdmini.dat` to suit your environment (see chapter 4.7 "Monitor Initial File `rdmini.dat`). The most important parameters in this file are: **TCPIPCV**, **X25RCV** or **SNARCV**, depending on the fact what kind of the network you use (TCPIP, X.25/ISDN or SNA). You are able to receive files, only if the value of this parameters equals 1 (default) or more. For more information see chapter 8 "rvs[®] Parameters".
16. Choose and define privileges according to your requirements (see chapter 3.2 "Choosing Privileges for rvsX") .

2.4 License key

After initial installation rvsX can be operated for test purposes without a license for a limited period of time. If you are satisfied with rvsX and would like to continue working with it contact your sales representative or rvs Support

Tel.: 030 / 399 71 777
 FAX: 030 / 399 71 994,
 E-Mail: rvs-service@gedas.de.

There you may obtain the desired license.

Update license key

- Start the program `rvsgmid` in the command line.
The program now displays a three- or four- character machine identification number on the screen.
- Inform your sales partner of the machine identification number and he will send you a license key by e-mail.
You should replace the old licence key (file `rdkey.dat`) in the `init` directory with the received license key. The new licence key must also be named `rdkey.dat`.
- Make a backup copy of the `RVSPATH/init/rdkey.dat` in order to be able to restore the original file status.

Structure of the license key data

0000000000	! Customer Number	00000010
X	! Product	00000020
211	! Release	00000030
5LTX	! Included Components	00000040
TESTINSTALLATIO	! Customer Name (Part	00000050
N	one)	00000060
	! Customer Name (Part	
	two)	
	! reserved	00000070
	! reserved	00000080
	! reserved	00000090
	! reserved	00000100
	! reserved	00000110
	! Computer Model	00000120
TESTVERSION	! PHYSICAL CPU	00000130
	IDENTIFICATION	
03211	! Expiration Date	00000140
CA580BE5 20189735 612035F9 77AD6591		00000150
162D0D7F 18FAB129 53B14EA4 79390F44		00000160
***** (C) gedas GmbH 2004		00000170
* make a success of rvs		00000180
*****		00000190

Included Components may contain two or more feature codes:

L LU 6.2 communications
T TCP/IP communications

X X.25 native / ISDN communications

Expiration Date is designated YYTTT where YY means the year, TTT the day-of-year of expiration.

3 Configuration of rvsX

This section describes how to customize station table and related tables for UNIX as well as which privileges you can choose for rvs[®] and how the network configuration and other basic configuration issues can be defined.

Which UNIX systems are supported, is described in paragraph "Short Description of the System". The differences with the configuration, which due to the different UNIX systems result, are explicitly characterized.

After you have installed the rvsX system, you must adapt rvsX to your request. Which possibilities of the configuration are available and how you can execute these, is explained in the following chapters.

3.1 Customizing Station Table and Related Tables

You need an rvs[®] entry in the station table for each partner station you want to communicate with.

The file

```
$RVSPATH2/rdstat.dat
```

contains a sample definition file for the station table and its related database tables that define the characteristics of the local station and of the other nodes you want to communicate with. Change this file according to your needs using an editor. This file will be used as default station table input file when the rvs[®] database is created.

After you have installed rvs[®] you will find a station table containing sample entries for X.25/ISDN, TCP/IP and LU 6.2 communication. Lines starting with (*) are interpreted as comments.

In general, the fields are case sensitive. When `rdstat.dat` is read, all field names and all those value fields that are not enclosed in simple quotes ('...') or double quotes ("...") are converted to upper case; e.g.

```
LUNAME=MYSTAT  
and  
luname=mystat  
are equivalent and different from both  
LUNAME="Mystat"  
and
```

² see chapter "Representation means" for the detailed description of `$RVSPATH`

LUNAME='mystat'.

Stations are identified and managed by Station IDs (SID) consisting of 16 characters. Each entry in the station table begins with a two characters table name. Line continuations are signalled by (+).The significance of the system tables is described in the following sections.

Next you are presented with a sample station table:

```
*****
**
* Define STATION-TABLE (ST), ROUTING-TABLE (RT), NACHBARKNOTEN (NK),
*
*       ODETTE-PARAMETERS (OP), APPC/LU62-PARAMETERS (LU)
*
*       X.25/ISDN-PARAMETERS (XP)
*
*****
**
*
* local station
ST SID=LOC NETID=??LOC?? STATNAME="local station"
RT SIDDEST=LOC SIDNEIGHB=LOC PRIORITY=1
NK SID=LOC FTP=O PROTOCOL=X PRIORITY=10
OP SID=LOC ODETTEID='my Odette ID'
LU SID=LOC LUNAME=XVWGGU01 NETID=VWAG TPNAME=RVSRCV USERID="" +
    PASSWORD="" MODE=VWG6RV10 SECURITY=0 +
    SYNCLEVEL=NONE TYPE=MAPPED
XP SID=LOC N=1 LINK=RVSLINEIN XADDRESS=05361268792 TIMEOUT=30
XP SID=LOC N=2 LINK=X25LINE XADDRESS=4553619301 TIMEOUT=300
TC SID=LOC N=1 INADDR=255.255.255.255 PORT=3305 MAX_IN=5
TC SID=LOC N=2 INADDR=255.255.255.255 PORT=2110 MAX_IN=10

* Partner Station - X.25 native communication sample
ST SID=RSX NETID=??RSX?? STATNAME='remote station 1'
RT SIDDEST=RSX SIDNEIGHB=RSX PRIORITY=1
NK SID=RSX FTP=O PROTOCOL=X PRIORITY=10
OP SID=RSX ODETTEID='my partners Odette ID' PSWFROM=XXX PSWTO=XXX
XP SID=RSX LINK=X25LINE XADDRESS=45536132200 TIMEOUT=300

* Partner Station - ISDN communication sample
ST SID=RSI NETID=??RSI?? STATNAME='remote station 2'
RT SIDDEST=RSI SIDNEIGHB=RSI PRIORITY=1
NK SID=RSI FTP=O PROTOCOL=X PRIORITY=10
OP SID=RSI ODETTEID='my partners Odette ID' PSWFROM=XXX PSWTO=XXX
XP SID=RSI LINK=RVSLINERSI XADDRESS=0536115303 TIMEOUT=30

* Partner Station - SNA-LU6.2 communication sample
ST SID=RSL NETID=??RSL?? STATNAME='remote station 3'
RT SIDDEST=RSL SIDNEIGHB=RSL PRIORITY=1
NK SID=RSL FTP=O PROTOCOL=L PRIORITY=10
OP SID=RSL ODETTEID='my partners Odette ID' PSWFROM=XXX PSWTO=XXX
LU SID=RSL LUNAME=LU62RSL NETID=VWAG TPNAME=RVSRCV USERID="" +
    PASSWORD="" MODE=VWG6RV10 SECURITY=0 +
    SYNCLEVEL=NONE TYPE=MAPPED
```

```
* Partner Station - TCP/IP communication sample
ST SID=RST NETID=??RST?? STATNAME='remote station 2'
RT SIDDEST=RST SIDNEIGHB=RSTPRIORITY=1
NK SID=RST FTP=0 PROTOCOL=T PRIORITY=10
OP SID=RST ODETTEID='my partners Odette ID' PSWFROM=XXX PSWTO=XXX
TC SID=RST INADDR=255.255.255.256 PORT=3305
```

```
* ROU (Routing sample: send file to station ROU via station RSX)
* ("via station" is defined in SIDNEIGHB)
* (RSX must have a partner with Odette-ID of ROU)
*
ST SID=ROU NETID=?ROU? STATNAME='example for routing via RSX'
RT SIDDEST=ROU SIDNEIGHB=RSX PRIORITY=1
OP SID=ROU ODETTEID='OROU' PSWFROM=aaa PSWTO=aaa
```

The different operation systems use the same station tables, differentiated only by the parameter **LINK**, which depends on the operation system. This parameter specifies the name of the used line definition for **AS/400**. For **Windows NT** and **UNIX** the parameter is pre-set in rvs[®]. For further information on the **LINK** parameter see section "X.25 native Communications XP".

The station table is read automatically, when the database is initialised. Later changes in the station table become effective only after deleting or re-initialising the rvs[®] database. Alternatively, you can type the command `modst` in the operator interface (see section 7 "Operator Console and Commands").

Note: Please observe, that the `modst` command adds new stations or changes the present ones.

If you want to delete stations, you can use the command `delst sid=...`

The related tables contain the information listed in the following sections.

3.1.1 Virtual Stations

rvsX supports virtual stations. Virtual stations are stations that represent stations that are not in the OFTP network. Virtual stations can be e.g. different users or applications. There is only one local station in rvsX, but you can have numerous virtual stations.

Files received for virtual stations are delivered locally. When sending a file, a virtual station can be set as a sender. For partner stations virtual stations are the stations that are reachable via routing.

To configurate a virtual station you should set only the ST and OP table. Especially important are the entry `SIDTYP=V` in the ST table, that indicates that a station typ is virtual and the `ODETTE-ID` in the OP table.

Example:

```
* VR1 (virtual station 1)
*
ST SID=VR1 NETID=?VR1? STATNAME='example for virtual station' SIDTYP=V
OP SID=VR1 ODETTEID='O0013003210GEDAS VR1'
```

In the station table of your partner your virtual station should be configured as a routing station that can be reached via your local station. Please see the example for a routing station in the chapter 3.1.1.

3.1.2 Identification of rvs® Nodes

In the station tables, there are several parameters that identify rvs® nodes:

SID This is a locally unique station ID which must consist of up to sixteen characters. This is a strictly local definition; remote stations do not have access to these names; they only know the ODETTE-IDs.

When choosing station IDs, keep in mind that these IDs will have to be used in all SendEntries (SE) and ResidentEntries (RE) to identify the target and the source of the transmission, respectively.

SIDTYP if the station is a virtual station, you should set the parameter SIDTYP=V.

OP.ODETTEID ODETTE ID in database table OP is a worldwide unique identification of all nodes that use the ODETTE File Transfer Protocol (OFTP). This 25 character name consists of

- the letter **O**,
- an 18 character organization identifier that is provided by the ODETTE codification group, and
- a 6 character computer subaddress that is administrated by each organization.

If you communicate within your own closed network only, the ODETTE ID may be freely chosen as long as it remains unique in your network.

In Germany, to apply for a ODETTE ID please contact :

Verband der Automobilindustrie e.V. (VDA)
 Abt. Logistik
 Postfach 17 05 63
 60079 Frankfurt
 Tel.: 069-7570-0

Get the complete description of OFTP from

<http://www.odette.org/>

3.1.3 Station Table ST

This system table contains information about your own station as well as information about all nodes that can be reached from the local station (either directly or indirectly, including the local station itself).

Table ST:

SID	Station ID; this name is unique within the local installation, only.
NETID	Unique ID in the entire rvs [®] network, do not implemented
STATNAME	Descriptive name of station This text will be displayed when a list of stations is requested in the rvs [®] dialog interface.
PHONE	Phone number of contact person at SID . This entry is a comment (optional).
SIDTYP	V if the station is a virtual station, you should set this parameter <code>SIDTYP=V</code> ; see chapter 3.1.1.

3.1.4 Routing Table RT

This system table defines for each rvs[®] station through which neighbor(s) it can be reached. If more than one neighbor could be used, the routes should have different priorities; the one with the highest priority at the time the `SK` command is created will be used.

The routing table must include an entry for your local station. Normally, the local station will be its own neighbor.

Table RT:

SIDDEST	Station ID of destination
SIDNEIGHB	Station ID of neighbor that offers a path to the destination
PRIORITY	Determines which path is selected (the smaller the numerical value, the higher the priority, i. e. <code>PR_HIGH < pr < PR_LOW</code>). Not yet supported.

3.1.5 Neighbour Node (NachbarKnoten) NK

This system table contains detailed information how (and when) other nodes can be reached.

Table NK:

SID	Station ID of neighbor
PROTOCOL	Line protocol Possible values: T TCP/IP (ODETTE Standard) X X.25 native / X.25 over ISDN (ODETTE Standard) L LU 6.2 R TCP/IP (rvs [®] internal only for UNIX) Default: T
FTP	File Transfer Protocol: O ODETTE Default: O
AUTODIAL	Determines whether rvs [®] automatically dials out if a data set is to be sent. Y a sender task will be started, as soon as a send request is available. N no sender task will be initiated, even when a send request is available. The queued data sets will be sent as soon as the partner establishes the connection, or an <code>activate sid=xxx</code> command for this partner is submitted from rvs [®] Console. Default: Y
DELAY	Time (in seconds) that rvs [®] should wait between connection attempts to this station. Default: 0
PSESSIONS	Maximum parallel sessions to be started to this partner. A value of -1 uses the global parameter MAXSESSIONS . Default: -1, use the global value from parameter MAXSENDERS . The overall number of parallel sessions is limited by MAXSESSIONS .

PRIORITY Determines which combination of **FTP** and **PROTOCOL** is selected (the smaller the numerical value, the higher the priority, i. e. $PR_HIGH \leq pr \leq PR_LOW$)

3.1.6 ODETTE Parameters OP

This system table contains ODETTE related information about all nodes (not just the neighboring ones!) that can be reached from the local station using this File Transfer Protocol. Valid passwords are needed for neighboring nodes only.

Table OP:

SID	Station ID
ODETTEID	As defined by ODETTE protocol (see section 3.1.2)
PSWFROM	Password that we are expecting this neighbor to send to us
PSWTO	Password that we are sending to this neighbor These ODETTE passwords are always exchanged and checked for neighboring nodes, independent of definitions for communication security as defined for LU 6.2.
SENDBLOCKS	Number of blocks to be sent before a checkpoint is reached. Use high values for "noiseless" connections to increase performance and low values for "noisy" connections. Default: 0 , use the value from global parameter SENDBLOCKS
RECVBLOCKS	Number of blocks to be received before a checkpoint is reached. Use high values for "noiseless" connections to increase performance and low values for "noisy" connections. Default: 0 , use the value from global parameter RECVBLOCKS
OEXBUF	ODETTE exchange buffer size in bytes Default: 0 , use the value from global parameter OEXBUF
OCREVAL	ODETTE FTP window size, number of buffers to be sent before waiting for a response Default: 0 , use the value from global parameter OCREVAL

CODEIN	<p>input code of the local file when creating a send enty</p> <p>A ASCII; E EBCDIC</p> <p>default: A (for ASCII)</p>
CODEOUT	<p>output code of the file to be send. Should be the local code of the remote host; on UNIX/NT systems it is A ASCII; on OS/400 and OS/390 E EBCDIC.</p> <p>default: X is LOCAL_CODE on the remote host.</p>
USERFIELD	<p>SPECLOG: Force OFTP special logic. NOT necessary under normal conditions; if a partner requests special logic, rvs[®] as a responder always answers with special logic. If USERFIELD is set to SPECLOG, rvs[®] as an initiator tries to establish the OFTP special logic. That means the OFTP packet SSID contains SSIDSPEC=Y. If the partner (responder) answers with SSIDSPEC=Y, the following packets are transmitted and received with a leading STX character, a block sequence character, the data and two trailing checksum characters.</p>
VDSNCHAR	<p>Range of allowable charactersto be transferred within an ODETTE transmission:</p> <ul style="list-style-type: none"> • ALL: no restrictions • OFTPUNIX: all capital letters, digits and the special characters . - • UNIX: all letters, digits and the special characters # _ - + . • ODETTE: all capital letters, digits and the special characters () - . / & • CHECK_RE: same as ALL, but it is necessary that a RE exists <p>default: ODETTE</p>
EERP_IN	<p>Wait for a receipt from the partner</p> <p>NEVER partner does not send EERP, so a send request ends with the correct transmission without waiting for the receipt</p> <p>NORMAL wait for receipt, end send request when receiving receipt</p> <p>Default: NORMAL</p>

EERP_OUT

Handling for sending a receipt

NEVER partner does not expect EERP, so don't create a receipt

IMMEDIATE create a receipt and start a session, if no session is available

NORMAL create a receipt, but wait for a session to transmit (suggested)

SYNC force transmission of a receipt (EERP) for a received file in the same session in which the file was received. The connection is not closed unless the EERP is created (after successful file delivery). The rvs parameter `SYNCDL` holds a time value in milliseconds that shall be waited before it is checked whether the EERP is ready to be sent. The number of wait periods is set by the rvsX parameter `SYNCTO`. The rvsX parameters `SYNCTO` and `SYNCDL` should be configured in `$RVSPATH/init/rdmini.dat`.

Example: Is `SYNCDL=400` and `SYNCTO=5`, a time span not exceeding 5 times 400ms is waited until the connection is closed. If in this time the EERP is created, it is transmitted, and the connection is closed immediately.

HOLD create a receipt, but do not send it. When a receipt is released, send it in the next session.

HOLD_IMMED create a receipt, do not send it. When a receipt is released, create a session and send it immediately.

Please read the chapter 3.1.7 for more information about releasing the receipts in the status **HOLD** or **HOLD_IMMED**.

Default: **IMMEDIATE**, recommended **NORMAL**

Note: If you do not want, that a receipt is sent immediately (with an eventual session creation), you should edit it (e.g. `EERP_OUT=NORMAL`) in the OP table of the station file (`rdstat.dat`).

ROUTING

Sometimes it is advantageous not to allow OFTP routing. This is possible by setting the rvs parameter `ROUTING` for single stations in the OP table. Using the same parameter in the file `$RVSPATH/init/rdmini.dat`, you can suppress/allow it or for all partner stations.

I (means IN): the incoming file transmission from the partner e.g. XXX to the remote partner e.g. REM1 via our local station e.g. LOC is permitted (XXX → LOC → REM1); not permitted is the outgoing routing e.g. for the partner REM2 via REM1 (LOC → REM1 → REM2).

O (means OUT): partner stations can't use your local station as router. Permitted is the outgoing routing e.g. for the partner REM2 via REM1 (LOC → REM1 → REM2). Not permitted is: the incoming file transmission from the partner e.g. XXX to the remote partner e.g. REM1 via our local station e.g. LOC (XXX → LOC → REM1).

B (means BOTH; IN and OUT): normal OFTP routing

N (means NEVER): routing in both direction IN and OUT forbidden.

default: **B**

SECURITY

This parameter specifies the usage of the data encryption. It can be set for the corresponding station here in the OP table or as rvsX global parameter in RVSPATH/init/rdmini.dat (then it is valid for stations without own SECURITY entry).

NO Encryption isn't possible. If a send job requests encryption the job is cancelled accompanied by an error message.

OPT Encryption is optional and may be set by each send job.(see the chapter 5.4.1 about Add.parameter)

FORCED Encryption is forced. If a send job does not switch on encryption a warning message is created and encryption is switched on. If the partner station sends an unencrypted file the reception is denied. A send job for a partner station is handled corresponding to the SECURITY entry for this station. It is not important whether a partner station is a neighbour station or is reached via routing.

More about encryption please read in chapter 9.

Default: **SECURITY=OPT**

3.1.7 How to release or delete an EERP in the HOLD or HOLD_IMMEDIATE status?

You can release or delete an EERP with the program `rvseerp`. This program is described in the Reference Manual. In this chapter only the options for deleting and releasing EERPs will be mentioned.

Usage:

```
rvseerp [-l] -r|-d -n <command number>
```

- The option `-l` lists all unended QSs (QuitungsSendung=EERP_OUT).
- `-r -n <command number>` releases the QS (EERP_OUT). `<command number>` is the command number of the QS. This command number will be listed with the option `-l`.
- `-d -n <command number>` deletes the QS (EERP_OUT). `<command number>` is the command number of the QS. This command number will be listed with the option `-l`.

Example:

```
rvseerp -l
```

Result:

```
# Log created at 3/28/2004 11:1
# QS: SIDORIG=AHM, SIDDEST=VS3, DTAVAIL=2004/03/28 10:57:31,
VDSN=TEST1 RETRY=0 STATUS=h
# rvseerp -r -n 129
# rvseerp -d -n 129
```

The result of the command `rvseerp -l` is detailed information about individual unended QSs such as:

- SIDORIG (SID ORIGINATOR, send station)
- SIDDEST (SID DESTINATION, target station)
- VDSN (virtual data set name, name for the file transmission)
- STATUS of the QS (e.g. h=hold)

The list command also writes (as comment) for each QS command a call of program `rvseerp` with the option `-r` (to release) or `-d` (to delete it). If you must release or delete numerous QSs, it is recommended to redirect the result of the `rvseerp -l` into a file.

Example:

```
rvseerp -l > eerpclean
```

After that you should edit the output file `eerpclean` and delete the comment character (`#`) before commands you want to execute. At the end you have to execute the edited file `eerpclean`.

If you don't have many commands to execute, you can call each command separately on the command line:

Example:

```
rvseerp -r -n 129
```

This command releases the QS in the hold status with the command number 129.

The QSs can also be deleted or released by the command line interface `rvsbat`.

Example (individual commands):

```
rvsbat
```

```
eerp/r cmdid=129
```

```
eerp/d cmdid=135
```

In the example you should call at first the program `rvsbat` on the command line and afterwards write the individual commands for release (`eerp/r`) or delete (`eerp/d`) the unended QS. The command number of the QS has to be set with the option `cmdid`.

EERP transmission can also be triggered in a Resident Receive Entry Script. The placeholder for the command number is `?CNQS?`. The EERP transmission is released by the `rvsbat` command `SEND /RELEASE CMDID=?CNQS?`.

Besides this the EERP may be released using the `rvs` CAL function `rvsQsRelease()`. (please refer to `rvscal.h`).

3.1.8 LU 6.2 parameters LU (only AIX, HP-UX and SINIX)

This system table contains LU 6.2 related information about the own local node or all neighboring nodes that can be reached from the local station using this protocol.

Table LU:

SID	Station ID; this name must be unique in the local installation. This is a required parameter.
PROFILE (only AIX)	LU 6.2 Partner Profile name. This is a required parameter. This is the name of the LU6.2 Partner Profile

and **LU6.2 Side Information Profile** in the SNA/6000 database. The name `rvscp` is default in the sample station table and in the sample LU6.2 profile (file `s_server.lis`).

LUNAME

LU name of remote host.

Default: one blank.

- **AIX:** This field has no effect and is only for your documentation. It can be left empty. The LU name has to be defined in the SNA/6000 profile.
- **SINIX:** For communication with rvsMVS, this is the VTAM APPL Macro definition for the XRLU defined in TRANSIT.
- **HP-UX:** This is a required parameter which can be defined in rvs[®].

TPNAME

Name of remote transaction processing program.

- **AIX:** This is an optional parameter. If this field is left empty, SNA/6000 uses the first name of the Remote Transaction Program List (RTPN list). Otherwise, the TPNAME must match the TP name of your partner station.
- For **HP-UX** and **SINIX** is this required parameter.
For your local station, this name must match the invocable TP name that the remote stations are calling (with `RVSRCV` as default). rvs[®] on an MVS-host (rvsMVS), uses `RVSOFTP` instead as its local TP name. If you intend to communicate with an rvsMVS make sure the host is calling you with TP name `RVSRCV` and you call it with `RVSOFTP`.

The following parameters are relevant only for HP-UX and SINIX systems

USERID

User ID to be used with remote program

Default: one blank

PASSWORD

Password needed to start remote program

Default: one blank

The required values for **USERID** and **PASSWORD** depend upon your communication partner's system and whether or not secure communication has been agreed upon.

- If your communication partner is another UNIX

node, turn communication security **off** on both sides when defining a Partner LU Profile.

- Sending: **USERID** and **PASSWORD** from your LU entry for the remote node are those that your neighbor defined in his list of User Profiles (outside of the Communication Manager).
 - Receiving: Your neighbor must enter those values for **USERID** and **PASSWORD** in his LU entry for your station that you defined in your list of User Profiles (outside of the Communication Manager).
- If your communication partner is an AS/400 node, turn communication security **on** when defining a Partner LU Profile.
 - Sending: **USERID** and **PASSWORD** should be set to empty strings **USERID="" PASSWORD=""** and the respective device at the AS/400 should be defined, so that it will start the receiver under a default user name and does not require a password.
 - Receiving: Your neighbor must enter those values for **USERID** and **PASSWORD** in his LU entry for your station that you defined in your list of **Dialog Security Profiles** (within the Communication Manager).

MODE

SNA session mode (one priority only)

Default: one blank

For communication with rvsMVS, this is the name of the VTAM Modetable Entry defined in the NCP LU Macro, with the **DLOGMODE** parameter.

We strongly suggest you select a mode name that is min. eight characters long, because some implementations of LU 6.2 do not pad shorter names properly! If this happens, the remote side may not recognize the mode name and refuse to establish a session.

SECURITY	specifies, whether password and user ID are expected: 0 (none) no user ID/password exchanged. This is the required value, if you communicate with rvsMVS. 1 (user security) userID/password must be specified explicitly Default: 0
SYNCLEVEL	APPC synchronisation level NONE no confirmation CONFIRM confirmation may be requested Default: NONE
TYPE	APPC conversation type BASIC basic conversation (not supported, at this time) MAPPED mapped conversation Default: MAPPED

Only **SINIX** systems require additional SNA related information. These parameters are stored in system table LX (for LU 6.2 extensions).

Table LX:

SID	Station ID; this name is unique in the local installation, only. This is a required parameter.
PUNAME	PU name of your physical unit as specified in VTAM/NCP PU statement and TRANSIT XPU-configuration. See also section 3.3. This is a required parameter.

3.1.9 X.25 native Communications XP

XP contains data of X.25-native communications.

Table XP:

SID	Station ID, defining local or remote station for which this entry is defined.
N	Key, to distinguish between different XP Blocks (e.g. to accept calls from different interfaces). Only supported for incoming calls. Use numbers beginning with 1.

- XADDRESS** DTE address, character string of maximum 15 decimal digits.
 For the local station, this is the own X.25 DTE address; for a remote station, it is the remote DTE address.
 Default: empty string
- TIMEOUT** Time period in seconds after which a connection is automatically terminated, if the partner station does not answer.
 Default: 60 (seconds)
- LINK** Link name, string of characters.
- X.25 adapter name. Standard for **AIX** is **x25s0** (IBM X.25 Coprocessor). Check your adapter name with `lsdev -C -H -t x25*`
 - If ISDN router BinTec BRICK is used: The link name for BRICK routers must be **RCAPI1** (**1** stands for the **control** field in the ISDN request **CAPI2_CONNECT_REQ** and can be changed, e.g. to vary the BRICK board number). See chapter "Defining the ISDN Network for BRICK Router" for more information.
 - **Only AIX:** If an ISDN board "diehl SCOM/2" is used together with "netISDN" software, the link name must be **NETISDN0**. If more than one ISDN board is used, a XP block for every board is required for the local station table entry. The second XP block must contain **NETISDN1** etc.
- ISDNNO** ISDN number of partner, string of characters.
 Required if ISDN connection is used (**LINK=NETISDN0** or **RCAPI1**). The sender will setup the ISDN connection to this partner, and then establish the X.25 protocol on this connection.
- FACILITIES** X.28 PAD profile string: If **FACILITIES** contains a string beginning with PAD, then the contents of this environment variable will be read and sent as a X.28 PAD profile string before OFTP SSRM packet. Example: If **FACILITIES=PADSTRING1** and the environment variable **PADSTRING1** contains 0230400D, then this 4 bytes will be sent as a PAD profile string before SSRM is sent.

USERDATA (only HP-UX and SINIX)

Call user data, string of maximum 256 hex characters (128 bytes); optional; default: empty string (no data)

Call user data are appended to the outgoing call packet and might be used by the remote installation as control information, especially for routing to the target application.

The first byte of the call user data field is commonly interpreted as "Protocol Identifier", PID, where some values are reserved, for example: X'C3' for SNA QLLC, X'C4' for SNA ELLC, X'EE' for TCP/IP.

Therefore, care must be taken when using call user data and there must be a mutual agreement with your partner station. Under normal circumstances, the call user data field is not required.

For incoming calls you can specify which protocol ID you want to accept with rvs[®], e.g. to distinguish between different partners. You can define 1 PID for each XP-Block. **USERDATA** is represented in hex, so for "RVS" you have to code "525653".

ALIAS

Optional IP address for ISDN router BinTec BRICK, format "nnn.nnn.nnn.nnn". When rvs[®] connects to a station with LINK=RCAP1 **ISDNNO=nnn ALIAS=nnn.nnn.nnn.nnn**, it establishes a TCP/IP connection to this IP address. If **ALIAS** is empty, rvs[®] reads the IP address from the UNIX environment variable **CAPI_HOST**. So, if you leave **ALIAS** empty, then define the variable **CAPI_HOST** in the rvs[®] profile (e.g. \$RVSPATH/.profile, see chapter "Representation means" for the detailed description of \$RVSPATH).

Defining **ALIAS** is especially useful if incoming calls originate from more than one BRICK router.

Additional for SINIX: string of characters. Provides a logical link to the CMX/CCP-WAN-X25 services.

For the local (own) station, this name must match with entry "name part[5]" in the TSNX definition for a local end system (Suggestion: X25_RVS_LOC).

For the remote station, this name must match with entry "name part[5]" in the TSNX definition for a remote end system (Suggestion: X25_RVS_RMT).

RECV_ALIAS string of characters. Provides a logical link for the rvs[®] receiver.

Additional for AIX: It is only necessary for the local (own) station. This name has to match with an entry in the **AIX** X.25 routing table (see AIX program "xroute").

The default is **IBMSAMP**. Sometimes another application uses the entry **IBMSAMP**, so you have to define another entry.

Additional for SINIX: It is only necessary for the local (own) station. This name must match with entry "name part[5]" in the TSNX definition for a local end system (Suggestion: X25_RVS_RCV). See also CCP-WAN definitions.

Receiving via X.25 and ISDN: If you want to be able to receive files from more than one partner at the same time, you have to prestart an X.25 receiver programs for each channel:

1. Edit the monitor initialisation file (`$RVSPATH/rdmini.dat`) and set the rvs[®] parameter **MAXX25RCV** to the number of receivers;
2. Edit the station table file (`$RVSPATH/rdstat.dat`) and add an X.25-parameter block ("XP") for each receiver to the local entry.
Only for **AIX**: Each receiver must have a different RECV_ALIAS name. See "Defining the X.25 Network for AIX" for more information.
3. Only for **AIX**: use program `xroute` to add routing entries.

Example for the local station definition (Receiving via ISDN BRICK router, listening on both ISDN channels, listen only for calls to our ISDN number with last digit=7):

```
XP SID=LOC N=1 LINK="RCAPI1" ISDNNO="7"
```

```
XP SID=LOC N=2 LINK="RCAPI1" ISDNNO="7"
```

Example for the remote station definition (call via BRICK ISDN router to ISDN no. 4711, optional X.25 address 20):

```
XP SID=R11 LINK="RCAPI1" ISDNNO=4711 XADDRESS=20
```

3.1.10 TCP/IP parameters TC

This system table contains TCP/IP related information about the own local node or all neighboring nodes that can be reached from the local station using the protocols **T** (ODETTE standard) and **R** (rvs[®] internal TCP/IP).

For installation, please refer to chapter "Defining the TCP/IP Connection".

Table TC:

SID	Station ID; this name is unique in the local installation, only. This is a required parameter.
N	Key which distinguishes between different TC blocks (e.g. to accept calls from different ports). For OFTP using TCP/IP, set: N=1 If you want to use the old rvs [®] internal TCP/IP, set: local station:N=0 remote station:N=0
PROTOCOL	Line protocol R TCP/IP (for rvsX only - old rvs [®] internal TCP/IP) T TCP/IP (ODETTE standard)
INADDR	Internet address / IP address or hostname Format 255.255.255.255 or rvsas1.gedas.de
PORT	Port to connect to (for OFTP is suggested to use 3305)
MAX_IN	Max. number of concurrent incoming transfers using the same port (relevant only for the local station).
MAX_OUT	Max. number of concurrent outgoing transfers using the same port (not yet implemented).

Example for the local station definition:

```
TC SID=LOC PROTOCOL=T N=1 INADDR="" PORT=3305
```

Example for the remote station definition:


```
TC SID=ABC PROTOCOL=T N=0 INADDR=xxx.xxx.xxx.xxx PORT=3305
```

3.1.11 Special Logic

In the rvs[®] station table now for each neighbour station can be determined if OFTP Special Logic usage is permitted (Error Recovery of OFTP Level 1.3 for rvs[®] is not yet supported). Special Logic can be configured by an entry in the station table (file `$RVSPATH/init/rdstat.dat`) like this example

```
LX SID=<SID> SPECIALLGC=Y|N
```

Default is N. The default behaviour is not to use Special Logic. With configuration **SPECIALLGC=Y** Special Logic is used, when the partner station agrees to that.

3.2 Choosing Privileges for rvsX

In order to exploit the full functionality of rvs[®], it is necessary that

- the Monitor can read all user data to be transferred;
- the Monitor can write received data into a user's directory, if required;
- the user can read data from the directory `$RVSPATH/usrdat/` (see chapter "Representation means" for the detailed description of **\$RVSPATH**) where all files are stored that are not explicitly directed elsewhere by means of resident receive entries;
- the Monitor can execute user jobs on behalf of a user because of a resident receive entry;
- the user or application can place orders into the rvs[®] database by means of the dialog, batch, or call interface.

On the other hand, the security control under UNIX allows to define privileges and access rules only on a rather crude level.

The two basic possibilities to run rvs[®], between which the customer has to choose, are described below.

3.2.1 Running rvs[®] high-privileged

The simplest way with respect to delivery of maximum functionality is to execute the rvs[®] Monitor under a user ID with **root** privileges. There will be no problem for the Monitor to read and write files and to execute user jobs out of a resident receive entry. The disadvantage is the potential abuse of the **root** privilege by rvs[®] users or the operator.

3.2.2 Running rvs[®] low-privileged

Using this mode, abuse of privileges is not possible. The disadvantage is that each rvs[®] user explicitly has to grant reading access to files to be transmitted and jobs to be executed out of a resident receive entry. If the **group** permission level is not applicable, there is only the **world** level left, which then allows access by everyone.

3.3 Defining the TCP/IP Connection

rvsX supports OFTP communication via TCP/IP. OFTP TCP/IP can be used for highspeed LAN connections as well as for internet connections.

The protocol OFTP TCP/IP is described in the internet documentation "RFC 2204". The recommended TCP/IP port is 3305.

3.3.1 rvsX Configuration for TCP/IP

Configure your rvsX:

Customize your station table (edit the file `$RVSPATH/init/rdstat.dat`) and your monitor initialisation file (`$RVSPATH/init/rdmini.dat`): See "Representation means" for the detailed description of **\$RVSPATH**.

Edit file `rdstat.dat`: Edit your own local station entry (in the default file, this is the station LOC). The TC line defines the TCP/IP receiver task. By default, leave the parameter **INADDR** empty. If you want to force rvsX to bind the TCP/IP listen socket on another IP address, you can set INADDR to this IP address. By default, rvsX listens on port 3305 (rvsX is able to handle multiple TCP/IP connections over the same port).

Then define a partner **STATION** with a TC line and an IP address. Set **INADDR** to the IP address of your partner. Set **PORT** to the OFTP-TCP/IP port number of your partner (default: **3305**).

Sample:

```
*****
*LOC (Definition of own local station)
*   (LINK=RCAPI1 ISDNNO="" means we accept all calls from BRICK)
*
ST  SID=LOC  NETID=LOC                      STATNAME='local rvs
station'
RT  SIDDEST=LOC      SIDNEIGHB=LOC  PRIORITY=1
NK  SID=LOC  FTP=O      PROTOCOL=T    PRIORITY=10
OP  SID=LOC  ODETTEID='O my ODETTE ID' +
      PSWFROM=AAA                      PSWTO=AAA
TC  SID=LOC  PROTOCOL=T    N=1  INADDR=""  PORT=3305
*****
```

```
*R11 (Definition of a partner station)
*
ST  SID=R11      NETID=R11                      STATNAME=
'OFTP-TCPIP partner'
RT  SIDDEST=R11  SIDNEIGHB=R11                  PRIORITY=1
NK  SID=R11      FTP=O  PROTOCOL=X    PRIORITY=10
OP  SID=R11      ODETTEID='O0013000001VW  R11'
      PSWFROM=AAA                      PSWTO=AAA
TC  SID=R11      PROTOCOL=T  N=0  INADDR=      PORT=3305
      xxx.xxx.xxx.xx
      x
*****
```

E.g., edit /home/rvs/init/rdmini.dat. Change:

```
setparm TCPIPRCV=1
```

This will start one TCP/IP receiver task. If you want to listen to more than one port, add additional TC lines to your local station definition and increase the parameter **TCPIPRCV**.

Then modify the rvsX database. Start the Operator Console (rvscns), and type

```
modst
```

The Operator Console should display the line

```
" I: <OK_CMD_DONE> [RVSCNS] 'modst' done. " .
```

Now you should be able to activate your partner with the operator command "ACT SID=xxx", for example:

```
act sid=R11
```

The Operator Console should display the line

```
I: <OK_ACTIVATE>    connect to station R11 ...
```

```
I: <CONNECTED>      Connection to station R11 established.
```

3.3.2 Problem Solving for TCP/IP

First, test the connection to your partner. A ping command should be successful:

```
ping xxx.xxx.xxx.xxx
```

Next, ensure that your partner station is listening on the defined TCP/IP port. For example, type

```
telnet xxx.xxx.xxx.xxx 3305
```

The partner should answer with the string "IODETTE FTP READY".

Next, ensure that your own station is listening on the defined TCP/IP port. For example, type

```
telnet own-ip-address 3305
```

Your rvs[®] should answer with the string "IODETTE FTP READY".

Next, you can check the rvsX TCP/IP listen socket with the command

```
netstat -a
```

You have to find a line like this:

```
tcp    0    0  hostname.3305  .          LISTEN
```

Be sure your partner connects to this hostname and port number.

3.4 Defining the ISDN Network for BRICK Router

rvsX supports OFTP ISDN Communication via an external ISDN router (BinTec BRICK).

BRICK has a "Remote CAPI" interface. That means, any computer in your LAN addresses the BRICK router as if it is an internal ISDN board.

A TCP/IP-based driver "tunnels" the ISDN packets to the BRICK router.

The rvsX OFTP software module addresses a BRICK router anywhere in the LAN/WAN of your company. That means:

1. The OFTP system does not need an internal ISDN board
2. Several OFTP systems (and other ISDN applications) can share the same BRICK router
3. Several routers in your WAN can provide multiple dial-in ports (fail-safe configurations)

BRICK is available for S0 (2-channel) or S2M (30-channel) ISDN.

gedas deutschland gmbh has tested the following routers:

- BIANCA/BRICK-XS
- BIANCA/BRICK-XM
- BIANCA/BRICK-XL
- X4100.

3.4.1 BRICK Router Installation

Install the BinTec BRICK Router in your LAN as described in the BRICK manual. For more information, visit the BinTec WWW homepage at <http://www.funkwerk-ec.com>.

Login on BRICK (telnet) und and choose "setup". The following mask appears:

```

          BIANCA/BRICK-XS Setup Tool
-----
Licenses                               System

LAN Interface:                         CM-BNC/TP, Ethernet
WAN Interface:                         CM-1BRI, ISDN S0
IP
ISDN Partner
    
```

Configuration Management

Exit

Choose "IP", "Routing". Your UNIX system must be inserted in the routing table.

Choose "License". Insert your BinTec BRICK license key.

Choose "ISDN", "Incoming Call Answering". Be sure that the local ISDN numbers which are used here, are NOT used for rvs[®] (Otherwise the BRICK may catch incoming calls which were intended for rvs[®]).

3.4.2 rvsX Configuration for BRICK Router

Configure your rvsX:

Define UNIX environment variables: Edit `/etc/profile` or `$RVSPATH/.profile` (see "Representation means" for the detailed description of **\$RVSPATH**). Insert your IP address:

```
CAPI_HOST=xxx.xxx.xxx.xxx; export CAPI_HOST
```

```
TRACE_HOST=xxx.xxx.xxx.xxx; export TRACE_HOST
```

Now login again to activate settings. On your rvs[®] system, type the **UNIX** command

```
ping $CAPI_HOST
```

It must be successful.

Customize your station table (edit the file `$RVSPATH/init/rdstat.dat`) and your monitor initialisation file (`$RVSPATH/init/rdmini.dat`). Then create the rvs[®] database and start rvsX:

Edit file `rdstat.dat`. Edit your own local station entry (in the default file, this is the station LOC). The 2 XP lines in the sample define 2 receiver tasks for "Remote-CAPI" (RCAPI) access.

The fields **ISDNNO** in your local XP lines define a "filter" for incoming calls. Incoming calls will be accepted only if the last digits of the calling address match this number (e.g. if you want to receive OFTP calls on number "123" and FAX calls on number "124", then set **ISDNNO=3**).

Then define a partner station with a XP line and an ISDN number.

Sample:

```
*****
*LOC (Definition of own local station)
* (LINK=RCAPI1 ISDNNO="" means we accept all calls
* from BRICK)

ST SID=LOC NETID= LOC STATNAME='local
rvs station'

RT SIDDEST=LOC SIDNEIGHB= LOC PRIORITY=1

NK SID=LOC FTP=0 PROTOCOL=X PRIORITY=10
OP SID=LOC ODETTEID='O my ODETTE ID' +
PSWFROM=AAA PSWTO=AAA

XP SID=LOC N=1 LINK=RCAPI1 ISDNNO=""
XP SID=LOC N=2 LINK=RCAPI1 ISDNNO=""
*****

*ZZZ (Definition of a partner station)
* (if you insert your ISDN number and your ODETTE ID,
* you can do a "loop test")

ST SID=ZZZ NETID=ZZZ STATNAME=
'looptest via
BRICK'

RT SIDDEST=Z SIDNEIGHB=ZZZ PRIORITY=1
ZZ

NK SID=ZZZ FTP=0 PROTOCOL=X PRIORITY=10
OP SID=ZZZ ODETTEID='O my ODETTE ID' +
PSWFROM=AAA PSWTO=AAA

XP SID=ZZZ LINK= RCAP11 ISDNNO=
00493039970813
*****
```

(You can define an additional X.25 call address or X.25 userdata field which may be needed by some partners; for example, define: **ISDNNO=123 XADDRESS=20 USERDATA=C0**).

Edit \$RVSPATH/init/rdmini.dat. Change:

```
setparm MAXX25RCV=2
```

This will start 2 X.25/ISDN receiver tasks which will wait for incoming ISDN calls. (A basic-rate ISDN line has 2 B-channels).

Then modify the rvs[®] database. Start the Operator Console (rvscns), and type

```
modst
```

The Operator Console should display the line

```
" I: <OK_CMD_DONE> [RVSCNS] 'modst' done. " .
```

Now you should be able to activate your partner with the operator command

```
act sid=ZZZ
```

The Operator Console should display the line

```
I: <OK_ACTIVATE> connect to station ZZZ ...  
I: <CONNECTED> Connection to station ZZZ established.
```

3.4.3 Problem Diagnosis for BRICK Router

First, test the connection to BRICK. A `ping CAPI_HOST` command should be successful.

You can analyse problems with the programs `bricktrace` and `capitrace`, which produce a `linetrace` output. BinTec also supplies a Windows95 based trace tool called "BRICKware DIME tools".

A simple tool to test your ISDN port is the program `rcapitest`. Just type for example

```
rcapitest receive $CAPI_HOST &  
rcapitest send $CAPI_HOST isdnno
```

where "isdnno" means the BRICK ISDN number. The "send" task will then communicate with the "receive" task.

Next, you can create rvs[®] log files. In the Operator Console (rvscns), type

```
setparm LITRACELVL=4
```

```
act sid=xxx
```

rvs[®] will now create log files (`rltr.log`) in the `temp` directory which log the BRICK connection commands.

3.5 Defining the ISDN Network for netISDN Base Software (only AIX Systems)

This chapter describes the configuration required to run rvs[®] with a diehl SCOM/2 board and netISDN software (from netCS GmbH). It is also possible to use the diehl S_{2M} ISDN board.

3.5.1 Adapter Installation for AIX

Preparation (as described in "netCS netISDN Installation Manual, Software Installation IBM AIX"):

1. Power off, insert SCOM/2 board in any slot, insert ISDN cable
2. Power on, login as root
3. Install the netISDN software from diskette

3.5.2 rvsX Configuration for AIX

Install rvsX as described. During the installation process, you will be asked:

Do you use network CAPI (e.g. BinTec Brick) for ISDN communication ?

Type Y (Yes).

Customize your stationtable (edit the file /home/rvs/init/rdstat.dat) and your monitor initialisation file (/home/rvs/init/rdmini.dat). Then create the rvsX database and start rvsX:

Edit /home/rvs/init/rdstat.dat. The first entry describes your own local station with the Station ID (SID), default value is LOC. Change 2 parameters:

```
OP SID=LOC ODETTEID='your_ODETTE_ID' +
XP SID=LOC N=1 LINK="NETISDN0" ISDNNO=your_isdn_number
```

Then change the entry of your partner (for example, SID "R40"):

```
NK SID=R40 FTP=0 PROTOCOL=X PRIORITY=10
OP SID=R40 ODETTEID='ODETTE_ID_of_partner' +
PSWFROM=partner_password PSWTO=your_password
XP SID=R40 LINK="NETISDN0" ISDNNO=partner_isdn_number
```

Edit /home/rvs/init/rdmini.dat. Change:

User Manual rvsX

```
setparm MAXX25RCV=1
```

This will start 1 X.25/ISDN receiver task which waits for incoming ISDN calls. For one ISDN adapter, you can start up to 2 receiver tasks (because one ISDN line has 2 B channels).

For one S_{2M} ISDN adapter you can start up to 30 receivers.

Then modify the rvsX database. Start the operator console (`rvscns`), and type

```
modst dsn="/home/rvs/init/rdstat.dat"
```

The Operator Console should display the line

```
" I: <OK_CMD_DONE>      [RVSCNS]  'modst' done. " .
```

Now you should be able to activate your partner with the operator command

```
act sid=R40
```

The Operator Console should display the line

```
I: <OK_ACTIVATE>      connect to station R40 ...  
I: <CONNECTED>       Connection to station R40 established.
```

If an ISDN error occurs, read the next section. If nothing happens, probably a library is not properly installed (`/usr/lib/libсна.a` and `/usr/lib/libx25s.a`). To check this, leave `rvscns` and type `rvscom /a R40`. This may result in the message `Could not load library libsna.a`.

You either can install the AIX SNA and/or X.25 package or use a different version of the `rvscom`.

3.5.3 Diagnosis for AIX

If an ISDN error occurs, first test the ISDN board installation with the program `rvsnetisdn`. Log in as root and type:

```
rvsnetisdn
```

You will see this menu:

```
RVS      netISDN utilities
```

1. Start netISDN

2. Stop netISDN
3. Show netISDN status
4. Show netISDN statistics
5. Start monitoring ISDN line
6. Show ISDN monitor output
7. Call myself (ISDN loop test)
8. Allow remote login
9. Allow remote copy
10. Remote login
11. Remote copy
- q. Quit

First, check the status of netISDN (topic 3). The status must be "RUNNING". Otherwise, start netISDN (topic 1). If you use EuroISDN (ETSI), the message has to be

```
Loading <TE_ETSI.SY>.....  
  
netISDN started ...
```

To test the ISDN connection, call your own number (topic 7). The program will send data via one ISDN B channel and will get it back via the second B channel. The test must end with the message "ISDNTEST: OK!". With this topic, you can test the connection to any other OFTP ISDN partner, too. Just type in the ISDN number of your partner.

The topic "Allow remote login" is useful to let your rvsX supporter log into your system.

The topic "Remote copy" is a simple filecopy function via ISDN, but it only works on systems with netISDN (**UNIX**) or "acopy" servers (MS DOS).

3.6 Defining the ISDN Network for internal ISDN board (only SINIX Systems)

The configuration required to run rvs[®] with the internal ISDN communication controller is described in the "Communications Manager SINIX V5.1 (CMX) Operation and Administration (SINIX, Reliant UNIX)":

http://www.siemens.de/servers/man/man_us/com_man.htm

To use OFTP over ISDN for the internal ISDN communication controller you have to do the following steps:

1. Configure the ISDN configuration (KOGS) in CCP / CMX
2. Configure TNS to use X.25 over ISDN
3. Configure FSS
4. Configure rvsX (SINIX)

3.6.1 Configuring TNS for SINIX

To configure the connection between rvsX (**SINIX**) and the ISDN communication controller you have to define the following TNS entries :

1. RVS_LOCAL for your local identification
2. For each partner RVS[®] "partnerid"

The definitions for TNS must match the definitions in rvsX (**SINIX**) and your FSS configuration.

3.6.2 Configuring rvsX (SINIX) rdstat.dat for ISDN communication

Customize stationtables, as described in chapter 3 on page 30.

For your own station you must add a XP definition as followed:

```
XP SID="own SID" N=1 XADDRESS="own ISDN no"  
RCV_ALIAS=ISDN_RCV ALIAS=RVS_LOCAL
```

For each partner you must add a XP definition as followed :

```
XP SID="partner SID" N=1 XADDRESS=USE_TNS  
RCV_ALIAS=RVS"partners SID"_ALIAS=RVS"partners SID"
```

- use command modst from rvs[®] console to update your rvs[®] station configuration.
- use act "partner SID" from rvs[®] console to test connectivity.

3.7 Defining the X.25 Network for AIX

This chapter describes the configuration required to run rvs[®] with X.25 native communication.

OFTP allows connections via a X.25 network like Datex-P or TRANSPAC.

3.7.1 Adapter Installation for AIX

Preparation:

1. Read the "AIXLink/X.25 Guide and Reference"
2. Install X.25 coprocessor board
3. Login as root
4. Install AIXLink/X.25 system software
5. With "smit" change network parameters (such as Network User Address)
6. Connect system and X.25 modem/router

3.7.2 rvsX Configuration for AIX

Customize your station table (edit the file \$RVSPATH/init/rdstat.dat) and your monitor initialisation file (\$RVSPATH/init/rdmini.dat): See chapter "Representation means" for the detailed description of \$RVSPATH.

- Edit file rdstat.dat
The first entry describes your own local station with the Station ID (SID) "LOC". The "XP" line defines the X.25 receiver task. The parameter **RECV_ALIAS** defines the entry for the AIX X.25 routing table (see below). The parameter **XADDRESS** defines your own X.25 network address. Then define a partner station with a "XP" line, the **XADDRESS** of your partner, and an optional **USERDATA** field.

Sample:

```
*****
* LOC (Definition of own local station)
*
  ST SID=LOC NETID=LOC           STATNAME='local rvs station'
  RT SIDDEST=LOC  SIDNEIGHB=LOC  PRIORITY=1
  NK SID=LOC  FTP=0  PROTOCOL=X   PRIORITY=10
  OP SID=LOC  ODETTEID='O my ODETTE ID' +
```

```
                PSWFROM=AAA                PSWTO=AAA
XP SID=LOC N=1 LINK="sx25a0"                RECV_ALIAS=IBMSAMP
XADDRESS="4512345"
*****
* R11 (Definition of a partner station)
*
  ST SID=R11 NETID=R11                STATNAME='OFTP X.25 partner'
  RT SIDDEST=R11 SIDNEIGHB=R11        PRIORITY=1
  NK SID=R11 FTP=0 PROTOCOL=X          PRIORITY=10
  OP SID=R11 ODETTEID='0001300001VW   R11' +
                PSWFROM=AAA                PSWTO=AAA
XP SID=R11 LINK="sx25a0"                XADDRESS=454711
*****
```

- Edit `$RVSPATH/init/rdmini.dat`
Change: `setparm MAXX25RCV=1`
This will start one X.25 receiver task. If you want to listen on more than one X.25 SVC, add additional XP lines to your local station definition and increase the parameter **MAXX25RCV**.

Then modify the rvs[®] database. Start the Operator Console (rvscns), and type

```
modst
```

The Operator Console should display the line

```
" I: <OK_CMD_DONE> [RVSCNS] 'modst' done. " .
```

3.7.3 X.25 Problem Diagnosis for AIX

Before investigating any problem, ensure that X.25 communications are set up correctly. Read the "AIXLink/X.25 Guide and Reference".

Use program `lsx25` to list the X.25 configuration. Use `xtalk` for a simple send and/or receive test. Use `xmon` to view X.25 packets.

3.7.4 X.25 Routing Information for AIX

For every incoming call, the operating system knows which application has to receive the call, because it holds a "routing table". The routing table contains "X.25 user data" filters, priorities and so on. You can change the routing table by using the program `xroute`.

rvs[®] uses the routing name `IBMSAMP` by default, so you don't have to change anything in the routing table. But, if you want to prestart more than one rvs[®] receiver (to be able to get more than one data set at the same time), you need additional routing entries. For example, to prestart 3 receivers:

1. edit the monitor initialisation file (`$RVSPATH/rdmini.dat`) and change the parameter **MAXX25RCV** to **3**.

2. edit the station table file (`$RVSPATH/rdstat.dat`) and add 2 X.25-parameter blocks ("XP") to your local station entry (each **RECV_ALIAS** name must be different).
3. run program `xroute` and add 2 routing entries.

Example:

```

XP      SID=LOC      N=1      LINK="sx25a0"      RECV_ALIAS=RVSA
XADDRESS="394710"

XP      SID=LOC      N=2      LINK="sx25a0"      RECV_ALIAS=RVSB
XADDRESS="394710"

XP      SID=LOC      N=3      LINK="sx25a1"      RECV_ALIAS=RVSC
XADDRESS="394711"

```

3.8 Defining the X.25 Network for SINIX

The X.25 Network Definition for SINIX is described in the "Communications Manager SINIX V5.1 (CMX) Operation and Administration (SINIX, Reliant UNIX)", section "Configuration in expert mode" manual. You find this manual in the internet:

http://www.siemens.de/servers/man/man_us/com_man.htm

3.8.1 rvsX Configuration for SINIX

Customize your stationtable (edit the file `$RVSPATH/init/rdstat.dat`) and your monitor initialisation file (`$RVSPATH/init/rdmini.dat`). See chapter "Representation means" for the detailed description of **\$RVSPATH**.

Edit file `rdstat.dat`: By default the first entry describes your own local station with the Station ID (SID) "LOC". The following sample refers to your own station.

Then modify the rvs® database. Start the Operator Console (`rvscns`), and type

```
modst
```

The Operator Console should display the line

```
" I: <OK_CMD_DONE>      [RVSCNS]  'modst' done. " .
```

3.8.2 Tracing X.25 packet level for SINIX

Under CMX, you can diagnose the connection in case of errors. You need good knowledge of X.25 packet structures. For complete information on CMX diagnostics, see "Communications Manager SINIX V5.1 (CMX) Operation and Administration (SINIX, Reliant UNIX)":

http://www.siemens.de/servers/man/man_us/com_man.htm

3.9 Defining the X.25 Network for HP-UX

This chapter describes a sample configuration with the necessary definitions required to run rvsX (**HP-UX**) with X.25 native communication.

3.9.1 Configuration of HP-UX for X.25

To run rvsX (**HP-UX**) with X.25, you must have installed X.25/9000 as described in

"HP-UX Installing and Administering X.25/9000"

In case of errors please refer to:

"HP-UX Troubleshooting X.25/9000" .

Configure

- your own DTE
- the name of the interface (link)
- other parameters

A sample configuration is shown below. Please adopt this example to your own needs.

```
#
#   X.25 Initialization File Created: Thu Aug 22 14:45:57 1996
#
# Global Parameters
#
x.121                <own DTE Addr>
x.121_packetaddr    <own DTE Addr>
device x25_0
name x25pgmaccess
#
# Level 2 Parameters
#
t1 3000
t3 60000
framesize 149
n2 20
l2window 7
#
# Level 3 Parameters
#
networktype DTE_84
#
# Circuit Table Definition
```



```

#
# LCI TYPE HOW MANY
lci 1 svc 17
#
# Flow Control, Throughput Class, Fast Select and Reverse Charge
Settings
#
flowcontrol off
thruputclass off
fast_select_accept disabled
reverse_charge disabled
def_inpacketsize 128
def_outpacketsize 128
def_inwindow 2
def_outwindow 2
def_inthruputclass 11
def_outthruputclass 11
#
# IP Parameters
#
ipaddress ????.????.????.??? 255.255.0.0
idletimer 600
holdtimer 300
mtu 2048

```

3.9.2 rvsX Configuration for HP-UX

Customize your station table (edit the file `$RVSPATH/init/rdstat.dat`) and your monitor initialisation file (`$RVSPATH/init/rdmini.dat`). See chapter "Representation means" for the detailed description of **\$RVSPATH**.

Edit file `rdstat.dat`: By default the first entry describes your own local station with the Station ID (SID) "LOC". The following sample refers to your own station.

Then modify the rvs[®] database. Start the Operator Console (`rvscns`), and type

```
modst
```

The Operator Console should display the line

```
" I: <OK_CMD_DONE> [RVSCNS] 'modst' done. " .
```

The "XP" line defines the X.25 receiver task.

Sample :

```
XP SID=LOC N=1 XADDRESS="*" USERDATA="" LINK=""
```

Use

- XADDRESS
- LINK
- USERDATA

for defining addressing options for the OFTP server.

Refer to "Addressing options for servers" in the "HP-UX X.25/9000 Programmers Guide" to define addressing options.

```
XADDRESS match X.121 Adress
```

```
LINK match X.25/9000 Interface name
```

```
USERDATA match Protocol ID
```

If you define **USERDATA** notice to use HEX codes.

Edit `$RVSPATH/init/rdmini.dat`. Change:

```
setparm MAXX25RCV=1
```

This will start the X.25 server task.

Then define a partner station with a "XP" line as described below.

Sample:

```
XP SID=XXX N=1 XADDRESS=xxx USERDATA="" FACILITIES=""  
LINK=""
```

Use

- **XADDRESS**
- **LINK**
- **USERDATA**
- **FACILITIES**

for defining addressing options for your communication client.

Refer to "Addressing options for clients" in the "HP-UX X.25/9000 Programmers Guide" to define addressing options.

```
XADDRESS match X.121 Adress
```

```
LINK match X.25/9000 Interface name
```

```
USERDATA match Protocol ID
```

```
FACILITIES match optional facilities
```

If you define USERDATA or FACILITIES notice to use HEX codes.

3.10 Defining the SNA Network for HP-UX

This chapter describes a sample configuration with the necessary definitions. To run rvsX (**HP-UX**) with LU6.2, you must have installed SNAplus as described in "HP-UX SNAplus Installation Guide". Configure the SNAplusLink as described in "HP-UX SNAplusLink Administrator's Guide".

Configure

- mode
- local LU
- remote LU
- local TP
- remote TP

as described in "HP-UX SNAplusAPI Administrator's Guide".

An example shows a running configuration. Please adapt this example to your own needs.

```
. *****
;
; SNAplus Binary to Text Configuration Utility
; Copyright (C) 1993 Hewlett-Packard Company
; Binary Configuration = /usr/lib/sna/com.cfg
; Security File        = /usr/lib/sna/com.sec
; File version         = 100.20
. *****
;
. *****
;
; Diagnostics Record (Mandatory)
. *****

[DIAGNOSTICS]
connection      = ""           ; Name of network
                                mgt connection
UCF_user        = ""           ; User ID for UCF
                                commands
error_log       = "/usr/lib/sna/sna.err" ; Error log file
audit_log       = "/usr/lib/sna/sna.aud" ; Audit log file
audit_level     = 10           ; Significant system
                                events
```

User Manual rvsX

send_overfl = No ; Send RTM when response counter max

send_end = No ; Send RTM at end of session

stop_timer = screen ; Data first reaches the screen

boundary_1 = 0.5 ; RTM histogram time boundaries

boundary_2 = 1.0

boundary_3 = 2.0

boundary_4 = 5.0

pc_error_log = "sna.err" ; PC client error log file

pc_audit_log = "sna.aud" ; PC client audit log file

. *****
;

; Local Node Record

. *****
;

[NODE]

name = "GEDANODE" ; Local Node Name

description = "NODE fuer Testinst." ; Description of Local Node

network = "" ; Node Network Name

. *****
;

APPC Local LU Record

. *****
;

[APPC_LOCAL_LU]

alias = "GEDASA01" ; LU Alias

node = "GEDANODE" ; Local Node

description = "Locale LU independent" ; Text description of LU

net_name = "NETZ" ; LU Network Name

LU_name = "GEDASA01" ; Name of LU

LU_number = 0 ; LU Number

```

session_lim      = 40           ; Session Limit
default_LU      = No           ; LU in pool of Default
                                   LUs
local_use       = No           ; LU can be used locally
partner_LU      = "RVSCICST, 2" ; List of Partner LUs and
                                   Modes
partner_LU      = "RVSRVS6, 2"
partner_LU      = "RVSRVSG2, 2"
partner_LU      = "RVSR11L, 2"

```

```

. *****
;
; SDLC Connection Record
. *****
;
[SDLC_CONN]
name             = "GEDACONN"   ; Name of connection
node            = "GEDANODE"   ; Name of node
description      = "Connection fuer ; Description
                                   BZO-Wolfs"
remote_end      = host         ; Remote end is host
activation       = initially    ; Initially active
XID_type        = 3            ; Format 3 XID
node_send       = "000.00000"  ; Node ID to send
node_rcv        = ""           ; Node ID to receive
control_point   = ""           ; Fully qualified
                                   control point name
encoding        = aaaa         ; Encoding is aaaa
full_duplex     = No           ; Full duplex
data_rate       = high;        ; Data rate is high
standby         = Yes          ; Standby
dial_data       = ""           ; Dial data
poll_address    = 00           ; Poll address
switch_timeout  = 00           ; Time to dial number
link            = "GEDASDLC"   ; link

```

. *****
;

; APPC Mode Record

. *****
;

[APPC_MODE]

name = "GEDASV10" ; Mode name
mode_ID = 2 ; Unique Mode ID
description = "APPC/LU6.2 ; Description
parallel Sess."
connection = "GEDACONN" ; Connection used by this
mode
priority = low ; Mode is Low Priority
session_limit = 10 ; Mode Session Limit
MCW = 5 ; Min Conwinner Sessions
partner_MCW = 5 ; Partner Min Conwinner
Sessions
auto_act = 0 ; Auto activated sessions
min_sendRU = 256 ; Min Send RU size
max_sendRU = 2048 ; Max Send RU size
send_pace = 16 ; Send Pacing count
min_rcvRU = 256 ; Min Receive RU size
max_rcvRU = 2048 ; Max Receive RU size
rcv_pace = 16 ; Receive Pacing count

. *****
;

; APPC Remote LU Record

. *****
;

[APPC_REMOTE_LU]

alias = "RVSCICST" ; LU Alias
description = "CICS-Test ; Text description of LU
DB/2"
net_name = "NETZ" ; LU Network Name
LU_name = "RVSCICST" ; Name of LU
SSCP_Alias = "RVSCICST" ; SSCP LU Alias
parallel_sess = Yes ; Parallel Sessions

```

supported
conv_sec          = No          ; LU uses conversation
                        level security
session_sec       = none        ; No Session Level
                        Security

. *****
;
; APPC Remote LU Record
. *****
;
[APPC_REMOTE_LU]
alias             = "RVSRVS6"   ; LU Alias
description       = "rvs_MVS    ; Text description of LU
                        Test"
net_name         = "NETZ"       ; LU Network Name
LU_name          = "RVSRVS6"   ; Name of LU
SSCP_Alias       = "RVSRVS6"   ; SSCP LU Alias
parallel_sess    = Yes         ; Parallel Sessions
                        supported
conv_sec         = No          ; LU uses conversation
                        level security
session_sec      = none        ; No Session Level
                        Security

. *****
;
; APPC Remote LU Record
. *****
;
[APPC_REMOTE_LU]
alias             = "RVSRVSG2"  ; LU Alias
description       = "rvs_MVS    ; Text description of LU
                        Test2"
net_name         = "NETZ"       ; LU Network Name
LU_name          = "RVSRVSG2"  ; Name of LU
SSCP_Alias       = "RVSRVSG2"  ; SSCP LU Alias
parallel_sess    = Yes         ; Parallel Sessions
                        supported
conv_sec         = No          ; LU uses conversation

```

```

level security
session_sec          = none          ; No Session Level
                               Security

. *****
;
; APPC Remote LU Record
. *****
;
[APPC_REMOTE_LU]
alias                = "RVSR11L"    ; LU Alias
description           = "R11 rvs-    ; Text description of LU
Produktion"
net_name             = "NETZ"        ; LU Network Name
LU_name              = "RVSR11L"    ; Name of LU
SSCP_Alias           = "RVSR11L"    ; SSCP LU Alias
parallel_sess        = Yes           ; Parallel Sessions
                               supported
conv_sec             = No            ; LU uses conversation
                               level security
session_sec          = none          ; No Session Level
                               Security

. *****
;
; 3270 Pool Record
. *****
;
[3270_POOL]
name                 = "gedas1"      ; Pool name
description           = "gedas1-LU-  ; Text description of pool
Pool"
model                = 2             ; Model 2 (24*80)
override             = No            ; User can change
                               screen model

. *****
;
; SDLC Link Record

```



```

. *****
;
[SDLC_LINK]
name           = "GEDASDLC"       ; Name of SDLC link
description    = "SDLC-Link der   ; Description
                Testinstallation"
device_name    = "sna_SDLC"       ; Name of device file for
                link
port_number    = 0                ; Adapter port
line_type      = leased           ; Leased
const_carrier  = Yes              ; Full duplex modem
                support

```

```

. *****
;
; Invocable Transaction Program Record

```

```

. *****
;
[INVOCABLE_TP]
alias          = "ORATEST"        ; TP Alias
description    = "Oracle test"    ; Text description of TP
load_method    = Q_auto           ; TP is queued,
                dynamically loaded
conv_sec       = No               ; TP needs user ID and
                password
TP_type        = APPC             ; TP is an APPC TP
file           = "/users/test/orac" ; Executable file name
TP_nametype    = char             ; TP name in characters
TP_name        = "ORATEST"        ; Full name of TP
parameters     = ""               ; Invocation parameters
environment    = ""               ; Invocation environment
target         = ""               ; Machine to load TP on
load_time      = 60               ; Timeout for loading TP
service_time   = 60               ; Timeout for servicing
                TP

```

. *****
,

; Invocable Transaction Program Record

. *****
,

[INVOCABLE_TP]

alias	= "RVSRCV"	; TP Alias
description	= "rvs Receiver LU6.2"	; Text description of TP
load_method	= auto	; TP is non-queued, dynamically loaded
conv_sec	= No	; TP needs user ID and password
TP_type	= APPC	; TP is an APPC TP
file	= "/users/rvs/system/r vscom"	; Executable file name
TP_nametype	= char	; TP name in characters
TP_name	= "RVSRCV"	; Full name of TP
parameters	= "/e/users/rvs/rvsenv .dat"	; Invocation parameters
environment	= ""	; Invocation environment
target	= ""	; Machine to load TP on
load_time	= 90	; Timeout for loading TP
service_time	= 60	; Timeout for servicing TP

. *****
,

; SDLC Link Usage Record

. *****
,

[SDLC_USAGE]

node	= "GEDANODE"	; Node name
link	= "GEDASDLC"	; Link name
incoming	= No	; Incoming calls accepted
encoding	= nrz	; Data encoding is nrz

```

. *****
;
; RJE FCB Record
. *****
;
[RJE_FCB]
name           = ".STDFCB"           ; Name of RJE Form
description    = "Default FCB"       ; Text Description
page_length    = 66                   ; Length of page (lines)
first_line     = 1                    ; First line output
last_line      = 66                   ; Last line output
channel_2      = 0                    ; Vertical Tab 2
channel_3      = 0                    ; Vertical Tab 3
channel_4      = 0                    ; Vertical Tab 4
channel_5      = 0                    ; Vertical Tab 5
channel_6      = 0                    ; Vertical Tab 6
channel_7      = 0                    ; Vertical Tab 7
channel_8      = 0                    ; Vertical Tab 8
channel_9      = 0                    ; Vertical Tab 9
channel_10     = 0                    ; Vertical Tab 10
channel_11     = 0                    ; Vertical Tab 11
channel_12     = 0                    ; Vertical Tab 12

```

3.11 Defining the SNA Network for AIX and SINIX

This chapter describes the Host Definition for **AIX** and **SINIX**, the SNA Server/6000 Definition for **AIX** and TRANSIT Definition for **SINIX** as well as the TRANSIT Support of Change Number of Sessions Verbs.

3.11.1 Host Definition for AIX and SINIX

In the following chapter an overview on how to code VTAM and NCP parameters is given. There is no guarantee for completeness and correctness. However, the coding examples given below originate from a functioning environment. For data security reasons, sensitive parameters like names, dialnumbers etc. have been changed. Please choose your own names for LUs, PUs, mode tables, log modes etc. and insert your own values for **DIALNO**, **IDBLK**, **IDNUM** and other parameters.

3.11.1.1 APPL Macro

The APPL Macro defines the HOST RVS as Major Node.

```
VBUILD          TYPE=APPL
RVSAPPL APPL    AUTH=(ACQ,PASS), PARSESS=YES, SPAN=(S
                P11),
                VPACING=7, MODETAB=LMTRVS, DLOGMOD=MT
                ERV10, ACBNAME=RVSAPPL,
                APPC=YES
```

3.11.1.2 Mode Table Definitions

The mode table defines the BIND parameters for LU 6.2.

```
LMTRVS
MODETAB
MTERV10          LOGMODE=MTERV10, BIND-Image for
ODEENT           LU6.2
                COS=BATCH1, TYPE=X'00',
                FMPROF=X'13', TSPROF=X'07
                PRIPROT=X'B0', SECPROT=X'B0', COMPROT
                =X'DOB1',
                RUSIZES=X'8686', PSERVIC=X'060200000
                00000000002300'
```

3.11.1.3 NCP BUILD Macro

The build macro has to include some parameters in order to support LU 6.2 PU type 2.1 connections.

```
NCPRVS01 BUILD
                ADDSESS=100,          LU 6.2 !
                AUXADDR=50,          LU 6.2 !
                BFRS=128,
                BRANCH=100,
                CA=(TYPE5,...),
                CANETID=(YXZ,...),
                CATRACE=(YES,128),
                COSTAB=ISTSDCOS,
                CWALL=32,
                DELAY=(.2,...),
                DR3270=NO,
                DSABLTO=3,
```

```

ENABLTO=20,
ERLIMIT=16,
HSBPOOL=500,
ITEXTTO=NONE,
LOADLIB=NCPLOAD,
LTRACE=4,
MAXSESS=20,          LU 6.2
MAXSSCP=6,
MAXSUBA=63,
MODEL=3725,
NAMTAB=200,          LU 6.2
NCPCA=(AVTIVE,...),
NETID=XYXYXY,
NETLIM=300,
NEWNAME=XXXXX,
NPA=(YES,DR),
NUMHSAS=60,
OLT=YES,
PATHEXT=100,
PRTGEN=GEN,
PWROFF=NO,
SESSLIM=255,
SLODOWN=12,
SUBAREA=33,
TGBXTRA=20,
TIMEOUT=(420,...),
TRACE=(YES,256),
TRANSFER=32,
TYPGEN=NCP,
TYP SYS=MVS,
UCHAN=NO,
VERSION=V4R3.1,
VRPOOL=250,
X25.MWINDOW=7,
X25.IDNUMH=03,
X25.MAXPIU=4000,
X25.MCHCNT=1

```

3.11.1.4 SDLC Leased Line Attachement

The line statement in NCP specifies the physical link for the attached units.

```
LNRVS00 LINE    ADDRESS= (nnn, HALF) ,
                CLOCKING=EXT,
                DATRATE=HIGH,
                DUPLEX=FULL,
                HISPEED=NO,
                MAXPU=1,
                NEWSYNC=NO,
                NPACOLL=YES,
                NRZI=YES,
                OWNER=SAnn,
                PAUSE=0.1,
                RETRIES= (7, 0, 0) ,
                SERVLIM=10,
                SPAN= (SP11) ,
                SPDSEL=NO,
                SPEED=9600,
                ISTATUS=ACTIVE
```

All necessary line configurations are depending on the physical line and modems, where the control unit (3174) is connected to the network controller (37xx / NCP). You can get these informations from the NCP systems administrator.

The PU statement specifies the physical control unit (3174).

```
PURVS00 PU      ADDR=C1,
                MAXDATA=265,
                MAXOUT=7,
                NPACOLL=YES,
                PASSLIM=7,
                PUTYPE=2,
                SPAN= (SP11) ,
                XID=YES,                ← (!)
                ISTATUS=ACTIVE
```

(!) The parameter **XID=YES** must be included for SNA-NODE 2.1 and Leased Lines! All other parameters for SNA-NODE 2.1 & LU6.2 must be included in the BUILD Macro Definitions for the NCP.

The LU statement specifies the Independent Logical Unit (LU) with parallel sessions.

```
LURVS00 LU      LOCADDR=00,                ← (!)
                DLOGMOD=MTERV10,
                MODETAB=LMTRVS,
                NPACOLL=YES,
                PACING=2,
                RESSCB=5,
                SSCPFM=FSS,
                SPAN=(SP11),
                VPACING=3,
                ISTATUS=ACTIVE
```

(!) The **LOCADDR=00** defines, that an Independent LU with parallel session Support is being used.

3.11.1.5 X.25 or Tokenring connections

X.25 or Tokenring are switched connections requiring definitions of a switched major node besides the Multichannel and Switched Line definitions for the physical attachment. Given below is a coding example for a switched major node for LU 6.2 PU 2.1 Attachment of rvs®.

```
VBUILD          TYPE=SWNET,MAXGRP=1,
                MAXNO=1

PURVS00         ADDR=C1,
PU              DISCNT=(YES,F),
                IDBLK=xxx,                ← (change !)
                IDNUM=xxxxxx,           ← (change !)
                IRETRY=NO,
                MAXDATA=261,
                MAXOUT=7,
                MAXPATH=1,
                PASSLIM=7,
                PUTYPE=2,
                SPAN=(SP11),
                ISTATUS=ACTIVE

PATH            DIALNO=4599999999,       ← (for X.25
                GRPNM=xxxxxxxx           dial-out)
```

```
LURVS00    LOCADDR=00,                ← (!!)  
LU         DLOGMOD=MTERV10,  
          MODETAB=LMTRVS,  
          PACING=1,  
          SSCPFM=FSS,  
          SPAN=(SP11),  
          VPACING=1,  
          ISTATUS=ACTIVE
```

(!) These parameters are used to localize the corresponding PU/LU upon dial-in. **IDBLK** is depending on the type of device attached. For example, an AS/400 system requires **IDBLK=056**, an OS/2 system requires **IDBLK=05D**. The **IDNUM** parameter may be chosen according to your local policy.

(!!) The **LOCADDR=00** defines, that an Independent LU with parallel session support is being used.

3.11.1.6 HOST Software Releases

The following Software Releases on an IBM Mainframe must be available to Support VTAM APPC and NCP with SNA 2.1 Definition :

VTAM Version 3.2	with PTF-Level	91.05
NCP Version 4.3.1	with PTF-Level	10.91
NCP Version 5.3	(FEP 3745)	

3.11.2 SNA Server/6000 Definition for AIX

LU6.2 connections are controlled by the subsystem SNA Server/6000. First, make sure that SNA Server/6000 is running on your system:

1. Check status of SNA server: smit.
2. Configure a session with: smit sna. We recommend to use the name `rvscp` for the LU6.2 Side Information Profile and LU6.2 Partner LU Profile so that this name corresponds with the `rvs®` stationable entry `PROFILE=rvscp`. One of the `rvs®` partner stations must be an `appn_network_node`. If your `rvs®` partner station is a MVS system, you can define your UNIX SNA system as an `appn_end_node`. You find a sample configuration in the file `/home/rvs/system/s_server.lis`.
3. Activate the link station.

4. Activate the session.

5. Check the Session: `lssrc -l -s sna`. The output should be like this:

Link station	Adjacent CP name	Node type	Device name	State	Number of sessions
@ent0			ent0	Starting	0
XXXXXX XX	XXXXXX.A BCD	EN	ent0	Active	10

6. Check rvs[®] connection: `rvscom /l rvscp`. The rvs[®] communication module rvscom tries to open the LU6.2 connection, allocates the remote transaction program (RVSOFTP) and waits for an OFTP Ready Message. If this test is successful, you can transfer files via rvs[®]. If it fails, use `smit sna - Diagnose`.

Here is a sample SNA/6000 profile. You can import it from `$RVSPATH/samples/sserver.lis` with `smit sna - Configure - Import` (see chapter "Representation means" for the detailed description of `$RVSPATH`). You have to change the marked parameters. Then verify with `smit sna - Configure - Verify`.

```
sna:
prof_name                = "sna"
max_sessions             = 200
max_conversations        = 200
restart_action           = once
rrm_enabled              = no
dynamic_inbound_partner_ = yes
lu_definitions_allowed
standard_output_device   = "/dev/console"
standard_error_device    = "/var/sna/sna.stderr"
nmvt_action_when_no_    = reject
nmvt_process
comments                 = ""
```

```
Control_pt:
prof_name                = "node_cp"
xid_node_id              = 0x071f0002 change!
network_name             = "DEIBMD1" change!
control_pt_name_alias    = "RVS2" change!
```

User Manual rvsX

```
control_pt_name           = "RVS2"      change!  
control_pt_node_type     = appn_network_node  
max_cached_trees         = 500  
max_nodes_in_topology_  = 500  
database  
route_addition_resistance = 128  
comments                 = ""
```

local_lu_lu6.2:

```
prof_name                 = "rvs"  
local_lu_name            = "LURVS002" change!  
local_lu_alias           = "LURVS002" change!  
local_lu_dependent       = no  
local_lu_address         =  
sscp_id                  = *  
link_station_prof_name   = "LINKRVS1"  
conversation_security_level_profile_name = ""  
comments                 = ""
```

partner_lu6.2:

```
prof_name                 = "rvscp" (rdstat.dat)  
fq_partner_lu_name       = "DEIBMD1.LURVS001"  
partner_lu_alias         = "LURVS001" change!  
session_security_supp    = no  
parallel_session_supp    = yes  
conversation_security_level = none  
comments                 = ""
```

side_info:

```
prof_name                 = "rvscp" (rdstat.dat)  
local_lu_or_control_pt_alias = "LURVS002" change!  
partner_lu_alias         = "LURVS001" change!  
fq_partner_lu_name       = ""  
mode_name                = "RVSMODE0"
```

```

remote_tp_name_in_hex      = no
remote_tp_name             = "RVSOFTP"
comments                   = ""
local_tp:
prof_name                  = "rvs"
tp_name                    = "RVSOFTP"
tp_name_in_hex            = no
pip_data_present          = no
pip_data_subfields_       = 0
number
conversation_type         = mapped
sync_level                 = none
resource_security_level   = none
resource_access_list_     = ""
profile_name
full_path_tp_exe          = "/home/rvs/system/
                           rvscom"           change!

multiple_instances        = Yes
user_id                   = 100
server_synonym_name       = ""
restart_action             = once
communication_type        = signals
ipc_queue_key             = 0
standard_input_device     = "/dev/console"
standard_output_device    = "/tmp/rvssna.out"
standard_error_device     = "/tmp/rvssna.err"
comments                   = ""
link_station_ethernet:
prof_name                  = "LINKRVS1"
use_control_pt_xid        = no
xid_node_id               = 0x071f0001    change!
sna_dlc_profile_name      = "rvs2"
stop_on_inactivity        = no
time_out_value            = 10
LU_registration_          = no
supported
LU_registration_profile_  = ""

```

```
name
link_tracing          = no
trace_format         = long
access_routing_type  = link_name
remote_link_name     = "RVS1"   change!
remote_link_address  = 0x000000000000
remote_sap           = 0x04
verify_adjacent_node = yes
net_id_of_adjacent_node = ""
cp_name_of_adjacent_node = "RVS1"   change!
xid_node_id_of_adjacent_ = 0x071f0001   change!
node
node_type_of_adjacent_ = learn
node
solicit_sscp_sessions = yes
call_out_on_activation = yes
activate_link_during_ = no
system_init
activate_link_on_demand = no
cp_cp_sessions_supported = yes
cp_cp_session_support_ = no
required
adjacent_node_is_     = no
preferred_server
initial_tg_number     = 0
restart_on_normal_    = no
deactivation
restart_on_abnormal_  = no
deactivation
restart_on_activation = no
TG_effective_capacity = 4300800
TG_connect_cost_per_time = 0
TG_cost_per_byte      = 0
TG_security           = nonsecure
TG_propagation_delay  = lan
TG_user_defined_1     = 128
TG_user_defined_2     = 128
```

```
TG_user_defined_3      = 128
comments                = ""

sna_dlc_ethernet:
prof_name               = "rvs2"
datalink_device_name   = "ent0"
force_timeout          = 120
user_defined_max_i_field = no
max_i_field_length     = 30729
max_active_link_stations = 100
num_reserved_inbound_  = 0
activation
num_reserved_outbound_ = 0
activation
dlc_protocol           = standard
transmit_window_count  = 16
retransmit_count       = 8
receive_window_count   = 16
inact_timeout          = 48
response_timeout       = 4
acknowledgement_timeout = 1
link_name              = "RVS2"    change!
local_sap              = 0x04
retry_interval         = 60
retry_limit            = 20
dynamic_link_station_  = yes
supported
trace_base_listen_link_ = no
station
trace_base_listen_link_ = long
station_format
dynamic_lnk_solicit_   = yes
sscp_sessions
dynamic_lnk_cp_cp_     = yes
sessions_supported
dynamic_lnk_cp_cp_     = no
```

```
session_support_required
dynamic_lnk_TG_          = 4300800
effective_capacity
dynamic_lnk_TG_connect_ = 0
cost_per_time
dynamic_lnk_TG_cost_per_ = 0
byte
dynamic_lnk_TG_security = nonsecure
dynamic_lnk_TG_         = lan
propagation_delay
dynamic_lnk_TG_user_   = 128
defined_1
dynamic_lnk_TG_user_   = 128
defined_2
dynamic_lnk_TG_user_   = 128
defined_3
comments                = ""
```

```
mode:
prof_name               = "rvs"
mode_name               = "RVSMODE0"
max_sessions           = 10
min_conwinner_sessions = 5
min_conloser_sessions  = 5
auto_activate_limit    = 2
max_adaptive_receive_  = 16
pacing_window
receive_pacing_window  = 7
max_ru_size            = 1024
min_ru_size            = 256
class_of_service_name  = "#CONNECT"
comments               = ""
```

3.11.3 SINIX TRANSIT Definition

The SINIX TRANSIT definition is described in the "Communications Manager SINIX V5.1 (CMX) Operation and Administration (SINIX, Reliant UNIX)" manual and in the "TRANSIT (UNIX) Documentation". You find these manuals in the internet:

http://www.siemens.de/servers/man/man_us/com_man.htm and
http://www.siemens/servers/man/man_us/tran_man.htm

3.12 Specify System Environment

rvs[®] derives the information where its various data sets are stored from a data set describing the local environment. The standard environment data set is found in

```
$RVSPATH/rvsenv.dat
```

which was created during the installation process of rvs[®]. The name of the path depends on the user which installed rvs[®]. This data set is pointed to by environment variable **RVSENV** which can be set in system file

- /etc/environment for **AIX**
- /etc/profile for **HP-UX, SINIX, Solaris, IRIX, Linux and SCO**

by means of the statement

```
RVSENV = $RVSPATH/rvsenv.dat; export RVSENV
```

The environment variable **RVSENV** can also be set in the local environment files of the user who has been installed rvs[®]. See chapter "Representation means" for the detailed description of **\$RVSPATH**.

This data set can be edited by the rvs[®] administrator and contains a set of parameters, which, besides other information, describe the directory structure of rvs[®]. More than one environment data set can exist. If for some reason you choose to run in another environment, you must call every rvs[®] main program with the '/e' flag immediately followed by the name of the environment data set containing the actually wanted definitions. This facility allows for maximum flexibility in the storage of rvs[®] data sets. This is convenient for example, if you want to install or test a new version of rvs[®] without disturbing operations of the productive rvs[®].

The environment data set contains parameters in the `name=value` syntax. Comment lines have to start with an asterisk (*) in column 1. Strings containing blanks, brackets, lowercase characters, and quotes have to be delimited by quotes or double quotes. Single quotes inside a string have to be doubled.

The environment parameters are:

AP_DISPLAY_INTERVAL	Interval in milliseconds, in which the display for active lines is updated Default: 1000
ARCDIR	directory for backup files Default: ARCDIR=\$RVSPATH/arcdir
DFTAUT	describes the default access rights to received files
MODE	The execution mode. Valid values are MODE=n for 'normal' execution or MODE=d for 'disabled'. The latter can be useful during database recovery. Default is N .
LANGUAGE	The language to be used if database is not available. Valid values are D for german and E for english dialog. Default is E for English.
DBLOG	specifies whether a database log file will be written during rvs [®] database accesses. If you specify Yes (Y), you can debug the rvs [®] database access. If you specify No (N), much disk space will be saved and performance is increased. Default: DBLOG=N
PATH	The common default path for all rvs [®] files. This specification can be overridden by the directory path definition described below. Default: \$RVSPATH \$RVSPATH depends on the system and can have different values: <ul style="list-style-type: none">• /home/rvs/ for AIX, IRIX, Solaris, Linux and SCO• /users/rvs/ for HP-UX• /defpath/rvs/ for SINIX If you choose to define a value for PATH other than the default value, all default directory names described below change accordingly.
DB	is the directory that contains all rvs [®] database and related files. Default: DB=\$RVSPATH/db/
INIT	is the directory containing all files that may be

	modified by the user like the stationtable or the initialization file for the Monitor. Default: INIT=\$RVSPATH/init/
SYSTEM	is the directory which contains all rvs [®] system files like programs, messages, help files and masks. Default: SYSTEM=\$RVSPATH/system/
TEMP	is the directory which will be used to store all temporary data sets. For example, incoming data will be stored in this directory during reception and will be deleted after successful delivery (copy) to the end user. Default: TEMP=\$RVSPATH/temp/
USRDAT	is the directory where the received data sets are copied to after successful transmission. Default: USRDAT=\$RVSPATH/usrdat/
SAMPLES	is the directory which contains shell scripts and C-Sources. See file header for explanation, and Reference Manual for a description of Batch and Call-Interface. This directory also contains a SNA-LU6.2 sample profile. Default: SAMPLES=\$RVSPATH/samples/
PORT	Port number of the server for messages of the line status Default: 2956
SERVER	Internet address of the server for messages of the line status Default: IP address or host name
KEYDIR	directory for private and public keys of local station and partner stations
SPINDIR	job input directory of Service Provider (SP); all files which should be converted concerning compression, encryption and code conversion are located in this directory; only SP internal, has no importance for user
SPOUTDIR	job output directory of Service Provider (SP); already converted files are located here; only SP internal, has no importance for user
SPFILESDIR	temporary work directory of Service Provider (SP); only SP internal, has no importance for user

3.13 Defining your Error Handling

Under rvs[®], you are able to specify the actions in case of errors.

If an error occurs (for example, a transmit error because of dropped connections), a log message will appear on the operator console. In addition, the shell script `$RVSPATH/system/rvserr` (see chapter "Representation means" for the detailed description of **\$RVSPATH**) will be executed, which may post the message as unix mail. So you don't have to look constantly for console messages to notice any problems.

If you don't want to get error mails, or if you want to specify your own error handling, you can edit the unix shell script `rvserr` located in path

```
$RVSPATH/system/
```

You can also write a C program with that name. See shell script `rvserr` for a description of parameters.

The compression and the encryption are realised by the program `rvssp`. This program stores the protocol files in the `$SPOUTDIR` directory and deletes them after the successful job execution. If an error occurs, the protocol files will not be deleted and can be analysed.

The environment variable `$SPOUTDIR` is defined in the file `rvsenv.dat`. More information about the file `rvsenv.dat` you can find in the chapter 3.12 "Specify System Environment".

4 rvsX Monitor

This chapter contains the description how to start and stop the rvs[®] Monitor for UNIX systems as well as how to set parameters and how to use rvs[®] Monitor commands.

4.1 Starting rvsX Monitor (`rvsstart`)

In this text, we assume that the Monitor is located in directory `RVSPATH/system/`.

Please, ask your system administrator, whether this is true for your installation and see chapter "Representation means" for the detailed description of **RVSPATH**.

`rvsstart` lets you start the rvs[®] Monitor as a background process:

```
rvsstart
```

This starts the Monitor with the command `nohup rvsmon&`. If there is a Monitor already running, an error message appears.

Switch to the proper directory, e.g. to `RVSPATH/system/`.

To start the monitor, simply say

```
nohup rvsmon&
```

or

```
rvsstart
```

and the Monitor starts running in background.

If `RVSPATH` is not part of your default access path as defined in the **PATH** system variable, switch to the proper directory by entering this command:

```
cd RVSPATH
```

To start the operator console, simply say

```
rvscns
```

and the Operator Console should start running. To stop the operator console hit the F3 key.

On the command input line you can enter additional inquiries, modify the way the Monitor executes or simply look what is going on.

4.2 Stopping rvsX Monitor (`rvsstop`)

To stop the rvs[®] system enter

```
stop
```

at the console input prompt [RVSCNS]

when using Operator Console (`rvscns`) or just use the command

```
rvsstop
```

`rvsstop` places a Monitor stop command into the rvs[®] database for either immediate or for delayed execution.

Note, however, that the stop command will only take affect, if Monitor is currently active (because a starting Monitor removes all `old' operator commands from the database). Therefore, it is better to include `opcnd cmd=stop time=1` in the Monitor initialization file to stop rvs[®] around 1 a.m.

Having stopped the monitor does not mean that rvs[®] is completely down, because sender and receiver tasks still can be active and even new receiver tasks could be started-up from remote. Also application programs using the batch interface or the dialog interface could still be active. If you want rvs[®] completely to shut down, close down any open dialog interface and rvs[®] using application, then disable external communication by shutdown

- any rvs[®] related communication daemon, e.g. the LU6.2 daemon. check that no rvs[®] task is running with command

```
ps -e | grep rvs
```
- TPStart program by closing that program icon in case of LU6.2 communication. Then check that no rvs[®] task is running.

If rvs[®] tasks are still running, the rvs[®] database may be destroyed or deleted, when you start backup or maintainace functions, so wait until all task has been ended.

Usage

```
rvsstop [/eRVSENV] [/lx] [/t] [/zhh[:mm[:ss]]] [/F] [/?]
```

All parameters are optional:

- /eRVSENV** specifies rvs[®] environment data set **RVSENV**.
- /lx** uses language **x** for prompts and messages.
- /t** stops Monitor executing in test mode.

- /z..** specifies time, when Monitor is supposed to stop; if that time has passed already today, the command will be scheduled for tomorrow.
- /F** stops Monitor immediately; all active senders and receivers are interrupted.
- /?** requests help information.

Example

```
rvsstop /e/home/sfr/rvstest/rvsenv.dat /ld /z3
```

Stop Monitor that uses rvs[®] database defined in environment data set /home/sfr/rvstest/rvsenv.dat; use German language for user communications. The Monitor is to stop at 3 a.m.

Use `stop xmt=force` to terminate Monitor, MasterTransmitter and all active Senders and Receivers immediatly. Active Senders will be interrupted abnormly. Exactly the same happens if you use the command

```
rvsstop /F
```

4.3 Stopping MasterTransmitter

MasterTransmitter `rvsxmt` controls

- the number of concurrently active Senders, depending on parameter **MAXSENDERS**
- the number of prestared "listening" receivers for X.25 native or ISDN communication, waiting to accept incoming calls, depending on parameter **MAXX25RCV** (provided your rvs[®] version is able to support X.25 native or ISDN).
- the number of prestared "listening" receivers for TCP/IP communication, waiting to accept incoming calls, depending on parameter **TCPIPRCV** (provided your rvs[®] version is able to support TCP/IP).

MasterTransmitter is started automatically during the Monitor's initialization phase.

When the Monitor is terminated with `stop` or `stop rvs=end`, the MasterTransmitter waits until all active senders has been ended, and then ends.

Use `stop xmt` or `stop xmt=end` to terminate MasterTransmitter after it stops all active Senders when `rvsxmt` receives such a stop command.

Use `stop xmt=force` to terminate Monitor, MasterTransmitter and all active Senders and Receivers immediatly. Active Senders will be interrupted abnormly. Exactly the same happens if you use the command

```
rvsstop /F
```

Upon closedown of `rvsxmt` all prestarted "listening" X.25 native, ISDN and TCP/IP Receivers will closedown too.

4.4 Killing rvs[®] Programs (`rvskill`)

`rvskill` lets you stop ("kill -9") the UNIX processes of rvs[®]:

```
rvskill
```

Then you will be asked which process of rvs[®] you want to stop. `rvskill` should be used ONLY if `rvsstop` does not work anymore (e.g. if the database is deleted or destroyed). If you are forced to use `rvskill` command call `rvsrii` afterwards (see the Reference Manual, chapter "Utilities"). If the database is damaged you should delete it (see chapter 6.5).

4.5 Using non default Database

Sometimes, it may be desirable to use a database other than the default one for rvs[®] operations. The path to the non-default database can be specified via an alternate environment data set specified as command line argument when the rvs[®] Monitor is started (the same is true for the other rvs[®] utilities like `rvsdia` or `rvsbat`, by the way). The environment data set, which by default is found as `$RVSPATH/system/rvsenv.dat`, contains all the necessary information for rvs[®] where to find the data sets and modules it needs for execution (see chapter "Representation means" for the detailed description of **\$RVSPATH**). But before this can happen, you have to create an alternate copy of the environment data set and edit it.

Let us assume that you want to start the Monitor with the database located in `$RVSPATH/system/DBnew/` to test some new line definitions without disturbing the production rvs[®]. Let us assume further, that the alternate environment data set is `$RVSPATH/system/testenv.dat`.

This alternate environment data set has to contain the following statement:

```
DB=$RVSPATH/rvs/system/DBnew/
```

To start the rvs[®] Monitor with the alternate environment, say

```
nohup rvsmon& /e/defpath/rvs/system/testenv.dat
```

Note, that there must not be any blanks between the option indicator `/e` and the start of the environment data set name. Note further, that this specification will not be remembered across sessions.

4.6 Setting rvs[®] Parameters at Start Time

Many characteristics of rvs[®] operation can be customized through rvs[®] parameters. Chapter "8 rvs[®] Parameters" contains a list of all parameter names and a brief description of their purposes. Chapter "8.2 rvs[®] Parameter Values" discusses some considerations how to choose rvs[®] parameter values.

Parameter values can be changed when the Monitor is started³. Note, that these changes are permanent and will be remembered across sessions.

Requests to set parameter values are entered as **NAME=VALUE** command line arguments; the number of these requests is only limited by the length of the command string that the system will accept.

Let us assume that you want to increase the priority of operator commands to **5** (the default is **10**) and change the rvs[®] Monitor suspension time⁴ to **1** minute (default is 30 seconds). To accomplish this, start the rvs[®] Monitor with

```
nohup rvsmon okprio=5 sleep=60 &
```

4.7 Monitor Initial File `rdmini.dat`

You may wish to have certain operator commands executed whenever the Monitor starts. This can be accomplished by

- storing those operator commands in file `$RVSPATH/init/rdmini.dat`, (see chapter "Representation means" for the detailed description of **\$RVSPATH**) and
- setting flag **/i** in the start command:

```
nohup rvsmon /i&
```

 (Since this flag is remembered across session, specifying **/i** actually is only necessary, after the flag has been turned off for some reason.)

`$RVSPATH/init/rdmini.dat` contains a sample definition file for Monitor initialization commands, i.e. those commands, that are executed whenever the rvs[®] Monitor is started (except if flag **/i0** is in effect). The default values of all parameters are shown as comments, if you remove the comment sign you can change this parameter to your own choice.

Edit this file according to your needs. Any operator command can be entered. See chapter 7 "Operator Console and Commands" for more information on operator commands.

³ Parameters may also be changed with the operator command `setparm` or in the Monitor initial command data set.

⁴ The Monitor suspension time is the period of time (in seconds) that the Monitor waits before looking for a new command when currently there is nothing to do for it. Longer periods decrease system overhead but make the Monitor less responsive to newly entered commands.

Use any editor to generate the initialization file. Please, follow these rules, when creating an initialization file:

- Any operator command may be specified.
- Each command must be contained in a separate, single line.
- Empty lines are ignored (so you can optically separate groups of commands).
- Lines starting with an asterisk (*) in column 1 are ignored (so feel free to enter comments or sample commands).

Sample Initialization File

A file that

- directs the Monitor to clean up the database by deleting all entries that completed more than three days ago; and
- makes sure that all line tracing is turned off,

might look like this:

```
*Sample Monitor initialization file
*(this is another comment line, followed by an empty line)

cleanup days=3 ss=yes
*turn all line tracing off

setparm      odtracelvl=0

sp           litracelvl=0
```

By default, the installation process stores a sample initialisation file with a commented description of all commands and parameters in the `$RVSPATH/init/rdmini.dat`.

4.8 Command Line Arguments

When starting the Monitor, a number of optional command line arguments may be specified.

They may be either

1. value parameters,
2. flags, or
3. assignment statements.

Value parameters and flags start out with a minus sign ("-") or a slash ("/"); they are case independent.

Note: All settings except the environment selection parameter **/e** and temporary flags are remembered across sessions!

Value Parameters

Value parameters expect a value to immediately (i. e. without intervening space) follow the parameter indicator.

/e specify an alternate environment data set containing alternate path information for rvs[®] standard data sets.

The value specified here affects only the current run; permanent changes require the default environment data set located in `$RVSPATH/system/rvsenv.dat` to be edited according to your needs (see chapter "Representation means" for the detailed description of **\$RVSPATH**).

Nota bene. This must be the first command line argument; otherwise, it may be ignored.

/l language for operator communication and LOG messages; default language for user communication. Must immediately be followed by language specification:

D Deutsch

E English

/w recreate database ("Wiederanlauf"). The fully qualified name of the old database log must be specified (this name must be different from the data set name that rvs[®] will be using for the current run).

flags may be followed by **0** or **1** to turn the flag **off** or **on**, respectively; specifying the flag name, only, turns the flag **on** (e. g. **/i1** is equivalent to **/i**). **/i** is the flag for reading initial commands.

Assignment Statements

Assignment statements are of the form **PARAM=VALUE**, with no spaces allowed. parm can be the name of any valid rvs[®] parameter.

4.9 Return Codes

The Monitor might return to the operating system with one of the following return codes:

0 normal termination

4 forced termination requested by operator

- 5** rvs[®] database is disabled
- 6** another Monitor is already active for the same database
- 99** sever internal error

5 How to Work Interactively with rvs

For UNIX systems you can use the dialog interface `rvsdia` to administrate and to receive and/or send files with `rvs`[®]. In addition, you have the Operator Console (see 7 "Operator Console and Commands") for the `rvs`[®] administration.

If you use the dialog interface `rvsdia`, the following functions are to your disposal:

- define send entries to transmit one data set at a time to one or more recipients.
- display the status of your transmissions.
- delete send entries, as long as `rvs`[®] did not start processing them, yet.
- create resident receive entries to influence the way incoming data is being stored or further processed.
- create job starts after send attempt in order to trigger actions when files are sent or couldn't be sent respectively.
- modify or delete resident receive entries and job starts after send attempt.

In the following sections, the masks (panels, screens, . . .) are shown as they appear under the UNIX operating system. Each mask is followed by a help text describing the purpose of the screen and the meaning of it's fields. These help texts are also available as online help (see function key **<F1>**, below).

5.1 Input and Output Fields

When you are asked to provide information to `rvs`[®], the panels will provide input fields which are separated by arrows (**===>**) from their descriptions; positions and maximum lengths of input fields are indicated by a series of underscores (_____) in the following sections.

Read only information is presented to you in output fields which are separated by colons (:) from their descriptions; positions and maximum lengths of output fields are indicated by periods (.....).

5.2 Function Keys

On most systems, you can use the following set of function keys to control the `rvs`[®] dialog interface. **Function key 1** will be called **<F1>** in this text, etc.

- <F1>** requests on-line help.
- A brief explanation of the input field, where the cursor currently is located, will be displayed. To

view the entire help text for the panel,

use function key **<F1>** to display context sensitive help, then press **<F1>** again; press **<ENTER>** to get to the next page; press **<ESC>** to leave help.

<F2> displays a mask to create a new entry (list displays).

<F3> exits panel, ignoring any input you may have made.

On systems that support an **Escape** key, **<ESC>** is equivalent to **<F3>**.

<F4> displays selection lists, if available.

In general, this key will work for fields, where you are asked to specify a station ID or a local data set name.

<F5> updates the current panel (when on-going transmissions are displayed) or confirms that you really want to perform an action (like deleting a send request).

<F7> scrolls up (list displays).

If available, **<PageUp>** is equivalent to **<F7>**.

<F8> scrolls down (list displays).

If available, **<PageDown>** is equivalent to **<F8>**.

Depending on the system and terminal type, not all function keys required could be mapped to keys indicated as **F..** keys on the keyboard. If you have problems with function keys, you can use the numerical keypad fields on your keyboard, instead.

If no function keys are available: The key

"?" works as **<F1>** (help),

"&" works as **<F2>** (add),

!" works as **<F3>** (exit),

"%" works as **<F5>** (delete/refresh),

"<" works as **<F7>** (up),

">" works as **<F8>** (down).

5.3 Dialog Interface (rvsdia)

The dialog interface can be invoked as

```
rvsdia [/e<envdsn>] [/l<language>] [/t]
```

where items within square brackets ([...]) are optional. The dialog interface can be used for administration and for sending and receiving tasks.

The command line parameters have the following meaning:

- **/e**: set alternate environment data set (for use by rvs[®] administrator only).
- **/l**: language: use message language given by character language
- **/t**: use test mode (for use by rvs[®] administrator only).

The dialog interface `rvsdia` starts out with the following initial mask:

RechnerVerbundSystem (mask INI)

```
-----
Rechner Verbund System
-----
RRRRRRR   VV   VV   SSSSSS
RR  RR   VV   VV   SS   S
RR  RR   VV   VV   SSS
RRRRRR   VV   VV   SSSS
RR  RRR   VV   VV   SSS
RR  RR   VVVV   S   SS
RR  RR   VV   SSSSSS

OPTION ==> █

1 - send a dataset
2 - display / delete transmissions
3 - delete send-requests
4 - resident receive entries
5 - send-job entries
6 - user entries
I - informations

| F1 = HELP, F3/ESC = EXIT
```

Select the action you want to perform next.

In most cases, follow-up panels will be displayed to ask for additional input or to display requested information. Active function keys are displayed in the bottom line and help you navigate the panels.

If you need help, press functionkey < **F1** > to get more information about each field in the displayed panel.

SELECT OPTION

Select the number of the option that describes the task you want to perform:

- 1 - send a data set: A panel will be displayed, where you can specify
- name of data set to be sent,
 - station ID and user ID of recipient,
 - special processing options.
- 2 - display / delete transmissions: You will be able to specify selection criteria for the display of transmissions (both sent and received).
- Select the transmission you are interested in
- to view details, (such as status, number of bytes, etc.)
 - cancel a send request
- 3 - delete send-requests: Choose this option if you know the parameters of the send request you want to delete; you will **not** see a selection list.
- 4 - resident receive entries: You will be able to specify selection criteria for the display of resident receive entries you want to work with.
- Select the resident receive entry (**RE**) you are interested in
- to view details,
 - to modify any field in the **RE**, or
 - to delete the entire **RE**.
- Through this selection you will be able to create a new entry too.
- 5 - job start after send attempt entries: You will be able to specify selection criteria for the display of job start after send attempt entries you want to work with.
- Select the job start after send attempt entry (**JS**) you are interested in
- to view details,
 - to modify any field in the

JS, or

- to delete the entire **JS**.

Through this selection you will be able to create a new entry too.

6 - user entries:

You will get a list of all rvs[®] users and you will be able:

- to view details,
- to modify a user entry,
- to remove a user entry,
- to create a new entry.

I - Informations:

This option provides access to additional on-line information about rvs[®].

5.4 Interactive Sending and Receiving

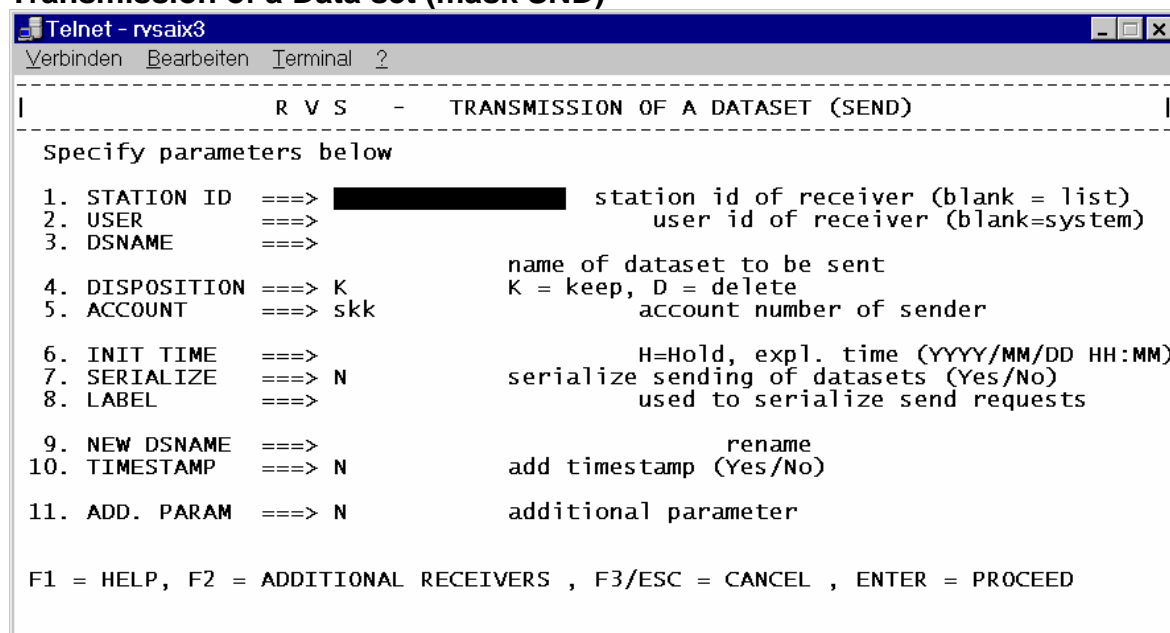
This chapter describes how to send and receive data sets with rvs[®] if you are using an interactive application `rvsdia`. First, it will be explained how you can create send requests. The next chapter describes how you can display sent and received transmissions and the last chapter shows how you can operate with send and receive data sets.

5.4.1 Create Send Requests

This chapter describes how you can create send requests and afterwards how you can transfer them.

Note: The size limit of the files to be transferred is at the moment 2 GB.

Transmission of a Data set (mask SND)



Specify the necessary parameters to create a send entry for transmitting a data set.

1. **STATION ID** Specify station ID of receiver's rvs® node.
 If you do not enter a value in this field, a list of available station IDs will be displayed.
 Use key < **S** > to select a station ID from that list and confirm your choice by pressing < **ENTER** >.
 Your choice will be inserted into the current panel.

2. **USER** Specify user ID of receiver
 An empty (blank) user ID indicates that the file should be delivered to the rvs® system at the target node, not to an individual user.
 Transmissions to MVS, VSE, or /36 systems must not specify a recipient's user ID, because these systems do not support

- person-to-person transfers. This is also true, if your file must be routed through one of these systems.
3. **DSNAME** Specifies the name of the data set to be sent.
The data set must exist. This field must have an entry.
4. **DISPOSITION** Availability of data set after sending
Choose:
 • **K** to keep data set after sending
 • **D** to delete data set after sending
default: **K**
5. **ACCOUNT** Specifies the account code of sender
Account code to which all activities relating to this send entry are to be charged.
default: account code of current session
6. **INIT TIME** Specifies the earliest time to send the data set
Choose:
 • **H** = Data set is put in held status; it will not be sent until released by you or freed by the rvs[®] operator.
 • **explicit time** = Year/Month/Day Hour:Minute (YYYY/MM/DD HH:MM)
 • **blank** = now
7. **SERIALIZE** Specify, whether this data set belongs to a group of serialized transmissions (see **LABEL** field)
Choose:
 • **Y(es)** for serialized sending of data set
 • **N(o)** for non-serialized sending of data set
default: **N**
8. **LABEL** Name of group of serialized send requests.
User specified (descriptive) label for this entry. It is used to serialize on another send entry with the same label, if serialization was requested (when there is

more than one other send entry with the same label, the latest one is used to serialize on).

9. NEW DSNAME

Rename data set for transmission.

This field specifies the name under which the file is known during transmission; resident receive entries at the destination must specify this name to further process the incoming data set.

If this field is left blank, the original data set name is used.

Note: Make sure to specify a valid MVS data set name when sending to an MVS-host!

10. TIMESTAMP

If you want to add timestamp:

Choose:

- Y(es)

default: **N**

11. ADD. PARAM.

You can add some another parameter such as FORMAT, INPUT CODE, OUTPUT CODE in the next mask if you

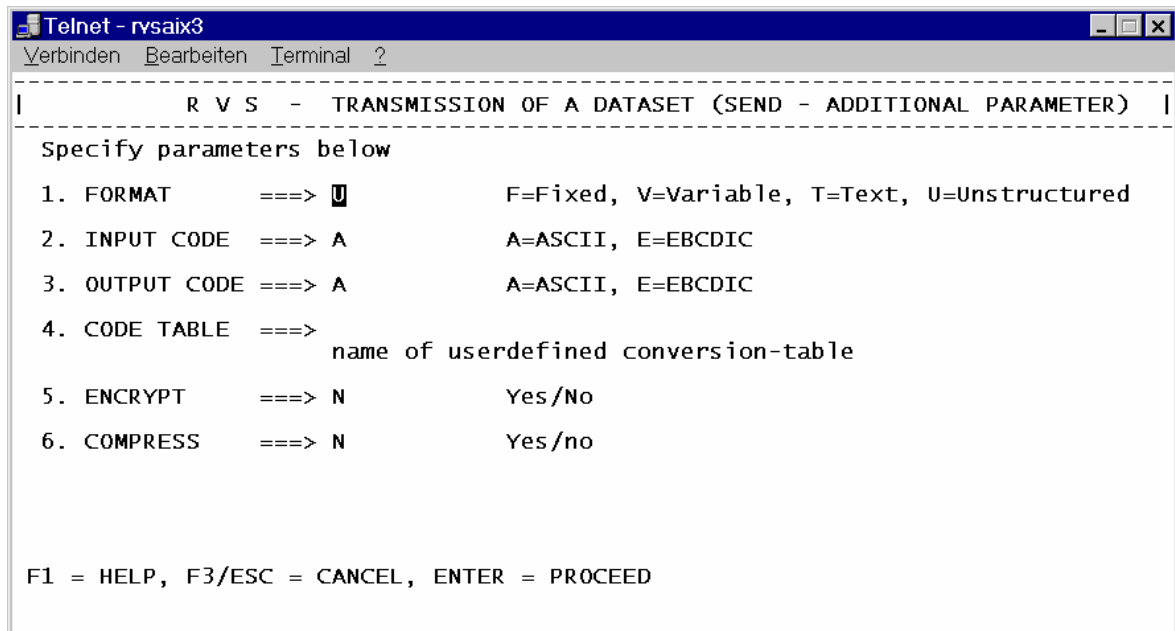
Choose:

- Y(es)

default: **N**

ADDITIONAL PARAMETER

This mask enables to you to set the additional send parameters.



1. FORMAT Specifies type of data set and format of transmission

Choose:

- **T** = text file; a stream of ASCII characters
- **U** = unstructured (binary) byte stream
- **V** = variable record length
- **F** = fixed record length
- **blank** = defaults to systemformat (e.g. **U** for rvsNT and rvsX, **F** for rvs400)

2. **INPUT CODE** Specifies the code in which the data set is currently stored
- Choose:
- **A** = ASCII
 - **E** = EBCDIC
 - **blank**:
for **T** format data sets, native code of system is assumed (EBCDIC under OS/400; ASCII, otherwise) for other formats, code is considered unknown and no conversion will take place
3. **OUTPUT CODE** Specify code in which data set should be delivered to recipient.
- By default, T-format data sets are delivered in the 'native' code of the receiver's system; non-text files are delivered without any conversion, unless you specify 'A' or 'E' in this field.
- Choose:
- **A** = ASCII
 - **E** = EBCDIC
 - **blank** = as explained in previous paragraph
- default: **blank**
4. **CODE TABLE** If you do not want to use rvs system code conversion tables, you can indicate your own code table with this parameter. You should write here the whole path of your code table. See chapter 10 for more information.
5. **ENCRYPT** If you want to send the encrypted files:
- Choose:
- **Y(es)**
- default: **N**
- See chapter 9 "Configuration of Encryption: Key Administration" for more information.

6. COMPRESS If you want to send the compressed files:

Choose:

- Y(es)

default: **N**

See chapter 9.2 "Configuration of Offline Compression" for more information.

Lists of Stations (mask SIDL)

```

-----
|                               R V S   -   LIST OF STATIONS                               |
-----
Option(s):  S - select

      Station-ID      Station name
-----
█ LOC              local rvs station
R10              aaa (X.25)
R11              VW AG (ISDN)
R20              aaa (LU6.2)
R30              aaa (TCP/IP)
R40              aaa (X.25 via ISDN)
RSX              remote station via X.25/ISDN
TEST            remote station via X.25/ISDN
ZZZ              echo test via X.25

-----
      F1 = HELP, F3/ESC = CANCEL, RETURN = PROCEED
Row 1 .. 9 - End of table reached

```

This mask is displayed, whenever you request a selection list of rvs[®] nodes that can be reached from the local station (either directly or indirectly).

Functionkey **<F3>** : Leave display without selecting any station ID

Functionkey **<F7>** : Page up.

Functionkey **<F8>** : Page down.

Key **<ENTER>** : Confirm selection.

Specify option If a station ID is selected by entering the line command **<S>**, the selected ID will be transferred into the appropriate field in the previous mask after you confirm your selection by pressing the **<ENTER>** key.

Send data set: Distribution List (mask SNDDL)

The list of currently defined recipients of your data set is displayed.

Use line command option **<S>** to select a receiver for modification, use **<D>** to delete a receiver from the list.

Functionkey < F2 >: The program displays a panel to define an additional receiver for this data set.

Functionkey < F3 > : Abort creation of the send entry; all information in the distribution list is discarded.

Functionkey < F7 > : Page up.

Functionkey < F8 > : Page down.

Key < ENTER > : Send data set.

Data set to be sent

Name of data set that should be delivered to one or more receivers.

Select option Enter an option in this column to modify or delete a receiver from the distribution list.

- Option **S**: Modify one or more parameters of this receiver.
- Option **D**: Delete this receiver from the distribution list.

STATION Station ID of receiver

USER User ID of receiver

OUTPUT CODE Code in which data set will be sent.

- **A** = ASCII
- **E** = EBCDIC
- **blank** = default

TIME STAMP Data set name should be made unique by adding a time stamp.

- **Y(es)** A time stamp will be added.
- **N(o)** No time stamp will be added.

NEW DSNAME Name in which the data set is to be delivered.

5.4.2 Display Send Requests and Received Transmissions

This chapter describes how you can display send requests and received transmissions.

Display / Delete Transmissions (mask DDT0)

```

-----
|                               R V S - DISPLAY / DELETE TRANSMISSIONS                               |
-----
Specify parameters below

1. DIRECTION => S R = receive, S = send
2. STATION ID => * id or pattern of remote station (blank=list)
3. USER      => * remote user id or pattern (blank=system)
4. LOCAL USER => * user id or pattern (blank=system)
5. DSNAME    => * blank or pattern for dataset selection list
6. DAYS      => * transmissions during last (up to 999) days

F1 = HELP , F3/ESC = CANCEL

```

Select one or more transmissions for display or deletion.

Functionkey < F3 > : Abort selection; return to previous panel.

Key < ENTER > : If station ID or data set name is blank, a selection list will be displayed. Otherwise, a list of matching entries will be shown.

DIRECTION Determines the direction of transmission.

Choose:

- **R**(eceive) display information about received files.
- **S**(end) display send entries.

default: both

STATION-ID Remote station ID.

Enter name or pattern of station ID of remote system.

If this field is blank, or <F4> is pressed while the cursor is positioned on this field, a list of available station IDs will be displayed. Use key <S> to select a station ID from that list and confirm your choice by pressing <ENTER>. Your choice will be inserted into the current panel.

USER User ID at remote system.

When you are looking at send entries, this is the receiver's name; for transmissions received at your station, this is the sender's user ID.

Enter name or pattern of user ID at remote system.

An empty user ID indicates that the file should be delivered to or was sent from the rvs[®] system at the target node, not to or from an individual user.

LOCAL USER

User ID or pattern of local user.

When you are looking at send entries, this is the sender's name; for transmissions received at your station, this is the recipient's user ID.

DSNAME

Name or pattern of local data set name.

A pattern may contain wildcard characters asterisk (*) and question mark (?). An asterisk matches any number of arbitrary characters, a question mark matches any one character.

If a pattern is specified, the list of transmissions will contain all matching entries.

If this field is blank, or **<F4>** is pressed while the cursor is positioned on this field, a list of available/matching data sets will be displayed. Use key **<S>** to select a data set and confirm your choice by pressing key **<ENTER>**.

This selected data set will be inserted in this field automatically.

DAYS

Specify, how far back the system should look for matching entries.

You can specify at most **999** days; the period of time, however, for which information about completed transmissions is kept in the system is determined by frequency and retention period for database cleanups.

<D>, <C>, and <H> can only be used as long as processing of send entry has not yet started.

<R> can only be used if request is held (status **h**)

New data set name

Name under which data set is known during transmission (**VDSN** = Virtual Data Set Name).

For single recipient, this is the name, under which it will be delivered.

SID

Station ID of receiver.

Date

Earliest point in time that this transmission should be executed, as defined when this entry was created (**month/day**).

Time

Earliest point in time that this transmission should be executed, as defined when this entry was created (**hour:minute**).

Status

Status of transmission at this time.

This field may show up to three statuses (e.g. **EEE** after processing is completed), namely status of:

- **SE**: command that controls processing of entire send entry; when it ends, the data set has reached its recipient.
- **ET**: description of one (local or remote) recipient (status can be **Q**, **E**, or **D**).
- **SK**: command that controls transmission to the neighboring node.

Commands may be in one of these statuses:

- **D** = deleted (by operator or user)
- **E** = ended (normal termination)
- **H** = set in hold status (by operator or user)
- **P** = pending (e.g. **SK** is waiting for receipt)
- **Q** = queued (waiting to be processed)
- **S** = suspended (all traffic to the neighboring node has been suspended by local rvs[®] operator) (**SK**, only)
- **I** = in transfer (ongoing transmission)
- **F** = forwardable (transmission may start)

Display of Transmission (send) (mask DSPT)

```

-----
|          R V S - DISPLAY OF TRANSMISSION (SEND)          |
-----
Cmdnbr of SE : 329                                DATE = 2000/04/03 TIME = 11:00
Cmdnbr of SK :
Local user   : umr
Account     : umr
Remote station: UMR                               / rvsstat-pcumr
Remote user  :
Dataset name : /home/umr/.profile
Format      : U
Dataset queued at ..... 2000/04/03 10:27
Earliest time to send (user) ... 2000/04/03 10:27
Earliest time to send (system) ..
Start of transmission .....
End of transmission .....
Number of transmitted bytes ..... 0                Blocksize 0
Number of attempts to send ..... 0

SE:queued ; ET:queued ; SK:-
Dataset will be kept after transmission
New dataset name is : TEST
F1 = HELP , F3/ESC = CANCEL , ENTER = REFRESH (for active transmissions)

```

The purpose of this panel is, to display a specified send transmission. As long as this transmission is active, you can refresh the status of transmission by pressing functionkey **<F5>**.

Functionkey **<F3>** : The program returns to the previous panel.

Functionkey **<F5>** : Refresh the status of active transmissions.

Cmdnbr of SE Internal number under which this send request is being processed.

You may need this number when communicating with your local rvs[®] operator (e.g. to have a send request deleted for which processing already started).

Cmdnbr of SK Internal number of the send command (SK) created from the send request (SE). Evaluates to **0**, if no SK has yet been created.

You may need this number when communicating with your local rvs[®] operator (e.g. to have a send request deleted for which processing already started).

DATE Current date

TIME Current time

Local user Local user ID of sender.

Account	Account to which all rvs [®] activities relating to this send request are being charged.
Remote station (ID)	Station ID of remote station.
Remote station (name)	Name of remote station.
Remote user	Remote user ID.
Data set name:	Name of data set to be transmitted.
Format	Format in which data set is sent. <ul style="list-style-type: none">• T = text file; a stream of ASCII characters• U = unstructured (binary) byte stream• V = variable record length• F = fixed record length• blank = Systemformat
Data set queued at (date)	Date of creation of this send entry (year/month/day)
Data set queued at (time)	Time of creation of this send entry (hour:minute)
Earliest time to send (user) (date)	Earliest point in time that this transmission should be executed as specified by the user when the entry was created. (year/month/day)
Earliest time to send (user) (time)	Earliest point in time that this transmission should be executed as specified by the user when the entry was created. (hour:minute)
Earliest time to send (system) (date)	Earliest point in time (system defined) that this transmission should be executed. (year/month/day) For 'queued' commands, this is the time when rvs [®] will attempt to execute the command; for 'pending' or 'ended' commands, this is the last time, when rvs [®] attempted to execute it.
Earliest time to send	Earliest point in time (system defined)

(system) (time)	that this transmission should be executed. (hour:minute)
Start of transmission (date)	Date when transmission started. (year/month/day)
Start of transmission (time)	Time when transmission started. (hour:minute)
End of transmission (date)	Date when transmission ended. (year/month/day)
End of transmission (time)	Time when transmission ended. (hour:minute)
bytes / blocks	Units (bytes or blocks) in which amount of transmitted data is measured.
Number of transmitted bytes/blocks	Total number of bytes or blocks transmitted up to now. This value is updated, whenever SEENBLOCKS buffers or blocks have been transferred (see rvs [®] Operator Manual for a description of parameter SEENBLOCKS).
Blocksize	Blocksize of transmitted file (zero for file formats T and U).
Number of attempts to send:	Total number of attempts (both successful and unsuccessful) to send file to neighboring rvs [®] node.
status line	Status of transmission will be displayed in line 20.
disposition line	Disposition of data set will be displayed in line 21.
new-dsn line	Virtual data set name (i.e. the name under which the data set is known during transmission) will be displayed in line 22.

Display of Transmissions (receive) (mask DDTR)

```
-----  
|                R V S - DISPLAY OF TRANSMISSIONS (RECEIVE)                |  
-----  
                                         DATE = 2000/03/29  TIME = 09:51  
Local user      : *  
Dataset name    : *  
  
Options : S-display additional information  
  
-----  
Dataset name (as received)  Station-ID (sender)      Date  Time  STATUS  
-----  
■  
  
-----  
F1 = HELP, F3/ESC = CANCEL, ENTER = REFRESH  
No Data - End of table reached
```

Information about receive entries is displayed.

Functionkey <F3> : Terminate and return to previous panel.

Functionkey <F5> : Refresh status of active receiver.

Functionkey <F7> : Page up.

Functionkey <F8> : Page down.

DATE Current date

TIME Current time

Local user Local user ID whose entries are shown.

Data set name Name of local data set.

Select option Option S: display additional information about this received data set.

Data set name (as received) Name of data set as received (virtual data set name).

Station ID of sender User ID of sender

Date Date, when data set was delivered.

Time Time, when data set was delivered.

STATUS Status of received data set.

This field may show up to three statuses (e.g. **EEE** after processing is completed):

- **IE**: command that controls processing of entire transmission; when it ends, a receipt for the sender is created.

- **ET**: description of one (local or remote) recipient (status can be **Q** or **E**).
- **IZ**: command that controls delivery to one local recipient.

Commands may be in one of these statuses:

- **D** = deleted (by operator)
- **E** = ended (normal termination)
- **H** = set in hold status (by operator)
- **P** = pending (e.g. **IE** may be waiting for all deliveries to complete)
- **Q** = queued (waiting to be processed)

Display of Transmission (receive) (mask DSPR)

```

-----
|                               R V S - DISPLAY OF TRANSMISSION (RECEIVE)                               |
-----
Cmdnbr of IE   : 337                               DATE = 2000/04/03   TIME = 11:06
Cmdnbr of IZ   : 338
Cmdnbr of RE   :                               Account in RE :
Creator of RE  :
Job ID of RE   :
Local user     :
Remote station: X3UM                               / RVSSTAT-aix3_umr
Remote user    :
Dataset name   : /home/umr/rvsxbr/usrdat/TEST2
Format        : U
Dataset queued at ..... 2000/04/03 11:05
Start of transmission ..... 2000/04/03 11:05
End of transmission ..... 2000/04/03 11:05
Dataset delivered at ..... 2000/04/03 11:05
Number of transmitted bytes ..... 4096           Blocksize  0
Number of attempts to receive ..... 0
Dataset has been completely delivered.
Dataset received as 'TEST2' / 'TEST2'
Dataset has been written to disk
F1 = HELP , F3/ESC = CANCEL , ENTER = REFRESH

```

The purpose of this panel is, to display a specified received data set. As long as this transmission is active, you can refresh the status of transmission by pressing functionkey **<F5>**.

Functionkey **<F3>** : The program returns to the previous panel.

Functionkey **<F5>** : Refresh the status of active transmissions.

Cmdnbr of IE Internal number under which this data has been received in your local system.

DATE Current date

TIME Current time

Cmdnbr of IZ Internal number under which this data set

	has been delivered to a local user.
Cmdnbr of RE	Internal number of resident receive entry, if this received data set has been processed by a matching resident receive entry
Account of RE	Account to which all rvs [®] activities relating receiving and storing a data set is to be charged, if resident receive entry is specified for this received data set.
Creator of RE	User ID of creator of resident receive entry, if specified for this received data set.
Job ID of RE	ID of batch job that further processed received information, if a resident receive entry is specified for this data set.
Local user	Local user ID.
Remote station (ID)	Station ID of remote station.
Remote station (name)	Name of remote station.
Remote user	Remote user ID.
Data set name	Local name of data set that has been delivered.
Format	Format in which data set is received. <ul style="list-style-type: none">• T = text file; a stream of ASCII characters• U = unstructured (binary) byte stream• V = variable record length• F = fixed record length• blank = Systemformat
Data set queued at (date)	Date of creation for sending of this data set on remote station. (year/month/day)
Data set queued at (time)	Time of creation for sending of this data set on remote station. (hour:minute)
Start of transmission (date)	Date when transmission on remote station was started. (year/month/day)
Start of transmission (time)	Time when transmission on remote station was started. (hour:minute)

End of transmission (date)	Date of end of transmission on remote system. (year/month/day)
End of transmission (time)	Time of end of transmission on remote system. (hour:minute)
Data set delivered at (date)	Date when data set was delivered on local system. (year/month/day)
Data set delivered at (time)	Time when data set was delivered on local system. (hour:minute)
bytes / blocks	Units (bytes or blocks) in which amount of transmitted data is measured.
Number of transmitted bytes/blocks	Total number of bytes or blocks transmitted up to now. This value is updated, whenever RECVBLOCKS buffers or blocks have been transferred (see chapter 8 "rvs [®] Parameters" for a description of parameter RECVBLOCKS).
Blocksize	Blocksize of transmitted file (zero for file formats T and U).
Number of attempts to receive	Total number of attempts (both successful and unsuccessful) to receive file from neighboring rvs [®] node.
status line	Status of transmission will be displayed in line 20.
Data set received as	Virtual data set name (i.e. the name under which the data set is known during transmission) will be displayed in line 21. There may be two different virtual data set names, if the sender sent the original data set to different users. In that case, the first VDSN is the label for the entire send request, and the second one is the VDSN used for delivery to you.
disposition line	Disposition of data set will be displayed in line 22, e.g. whether the data set has been ignored or written to disk.

5.4.3 Modify Send Requests and Received Transmissions

This chapter describes how you can modify data requests with rvs[®] if you are using an interactive application. It will be explained how to delete send requests.

Delete Send Request (mask DELSR)

```
-----  
|                               R V S - DELETE SEND REQUEST                               |  
-----
```

To delete send requests, specify the following parameters:

1. SID ==> | ██████████ station id of receiver (blank: disp. list)
2. USER ==> user id of receiver
3. DSN ==> blank or pattern for dataset selection list

F1 = HELP , F3/ESC = CANCEL , ENTER = PROCEED

In this panel, you specify a send entry that you want to delete. All fields must be specified.

Functionkey <F3> : Cancel delete request; no send entry will be deleted.

Key <ENTER> : When all fields have been specified, <ENTER> will check, whether such a send entry exists and can still be deleted (because it has not been processed yet). If the entry is available, a delete / confirm panel will be displayed.

If **SID** or **DSN** is still empty, a selection list will be shown.

SID Station ID of receiver

If this field is blank, a list of available station IDs will be displayed. Use key <S> to select a station ID from that list and confirm your choice by pressing <ENTER>. Your choice will be inserted into the current panel.

USER User ID of receiver

DSN Name of data set
Enter name of data set or blank or pattern
for data set selection list

Confirm Delete of Send Request (mask CDSE1)

You are asked to confirm that you really want to delete the selected send entry.

Functionkey <F3> : Cancel delete request; the send entry will
be kept.

Functionkey <F5> : The displayed send request will be deleted
and the file will not be transmitted.

Station ID of receiver Station ID of receiver.

User ID of receiver User ID of receiver.

Data set name Name of data set to be sent.

List of Data sets for Transmission (mask DSNL)

This panel lists all available (local) data sets for transmission. It is displayed whenever you request a data set selection list. Select a data set to include its name in the previous panel.

Functionkey <F3> : Leave this list without selecting a data set
and return to the previous mask.

Functionkey <F7> : Page up.

Functionkey <F8> : Page down.

Key <ENTER> : Confirm selection.

Specify option If a data set name is selected by entering
the line command <S>, the selected name
will be transferred into the appropriate field
in the previous mask after you confirm your
selection by pressing the <ENTER> key.

Local DSN Name of the local data set.

5.5 Interactive Administration

This chapter describes how to administrate resident receive entries, jobs after send attempts and user lists as well as how to call for information of rvs®.

You must call the application `rvsdia` if you want to work interactively with rvs®.

5.5.1 Resident Receive Entries

This chapter describes how to operate resident receive entries, how to start a job after send attempt, how user lists can be administrated and how you can get informations about rvs®.

Resident Receive Entries (mask RE)

R V S _ DISPLAY / UPDATE / DELETE RESIDENT RECEIVE ENTRY

To display/change/delete a resident receive entry specify parameters:

1. SID ==> XXXXXXXXXX station id of sender (name or pattern)
2. USER ==> * user id of sender (name or pattern)
3. DSN ==> * name or pattern of dataset as received (blank: all)
4. LOCAL USER ==> * user id of local user
(name or pattern, privileged users only)

F1 = HELP , F3/ESC = CANCEL , ENTER = show list of entries

The purpose of this panel is to display, change, or delete an existing resident receive entry. Specify the following parameters to describe the resident receive entry you want to access.

Functionkey **<F3>** : Cancel request; all receive entries remain unchanged.

SID Station ID of sender
If this field is blank, a list of available station IDs will be displayed. Use key **<S>** to select a station ID from that list and confirm your choice by pressing **<ENTER>**. Your choice will be inserted into the current panel.

This is one of the selection criteria; incoming data set must have originated at a station with a matching name for this receive entry to be further considered as a possible action for the incoming file.

To be actually used, all selection criteria must match.

USER User ID of sender
Name or pattern of user ID of sender.

One of the selection criteria; incoming data set must have originated from a sender with a matching name for this receive entry to be further considered as a possible action for the incoming file.

To be actually used, all selection criteria must match.

DSN

Name of received data set

Enter name of data set of resident receive entry. Leave blank or specify pattern to get a data set selection list.

One of the selection criteria; incoming data set must be known by a matching name for this receive entry to be further considered as a possible action for the incoming file. To be actually used, all selection criteria must match.

LOCAL USER

Name or pattern of local user ID who is the recipient of the incoming information.

You may specify something other than your own user ID, only if you are defined as privileged user.

Default: your current user ID

List of Resident Receive Entries (mask REL)

```

-----
|      R V S  -  DISPLAY / UPDATE / DELETE RESIDENT RECEIVE ENTRY      |
-----
                                DATE = 2000/03/29 TIME = 09:54
Options : S - display entry, U - update entry, D - delete entry
Dataset name          StationID (sender)          Action
-----
| | *                  *
-----

F1 = HELP, F2 = NEW ENTRY, F3/ESC = EXIT
1 row - End of table reached

```

This panel displays a list of resident receive entries.

Use line command **<S>**, **<U>**, or **<D>** to get more information about the entry, update, or delete it, respectively.

Functionkey <F1> : Display help information.
Functionkey <F2> : Display a mask to add a new entry.
Functionkey <F3> : Ignore any selection and leave display.
Functionkey <F7> : Page up.
Functionkey <F8> : Page down.
Key <ENTER> : Confirm selection.

DATE Current date

TIME Current time

Specify option Enter one of these options to the left of one of the displayed resident receive entries:

- **S**: Display resident receive entry.
- **U**: Update resident receive entry.
- **D**: Delete resident receive entry.

DATA SET-NAME Name or name pattern of incoming data set that will be processed by this resident receive entry.

SID Station ID of sender.

UID (sender) User ID of sender.

Local User Name or pattern of local user ID who is the recipient of the incoming information.

Action Indicates last action you performed against the list entry (**UPDATE**, **DELETE**) in the current session.

Create Resident Receive Entry (mask CRRE)

```

-----
|                R V S  -  CREATE RESIDENT RECEIVE ENTRY                |
-----
Specify paramters below

1. STATION ID  ==> * [REDACTED] station id of sender
2. USER       ==> *                user id of sender
3. LOCAL USER ==> *                user id of receiver
4. DSNAME     ==> *                dataset name
5. DSNNEW     ==>                only if to be renamed
6. ACCOUNT    ==> bwa                account number
7. TIMESTAMP  ==> N                add timestamp (Yes/No)
8. DISP       ==> K                disposition (Keep/Delete)
9. REPLACE    ==> N                replace dataset (Replace/New name/Ignore)
10. JOB       ==>                automatic job submit after receiving:
                                specify complete dataset name
                                containing job to be submitted.
11. COMMENT   ==>                comment concerning resident receive entry
F1 = HELP , F3/ESC = CANCEL , ENTER = create entry

```

In this panel, you can create a resident receive entry.

In a resident receive entry, a user may specify what actions should be taken, when a data set has been received.

When changes become necessary, a new entry will be created and the old one will be deleted.

Functionkey **<F3>** : No resident receive entry will be created.

Key **<ENTER>** : If all fields are specified the resident receive entry will be created.

STATION ID

Station ID of sender

If this field is blank, a list of available station IDs will be displayed. Use key **<S>** to select a station ID from that list and confirm your choice by pressing **<ENTER>**. Your choice will be inserted into the current panel.

This is one of the selection criteria; incoming data set must have originated at a station with a matching name for this receive entry to be further considered as a possible action for the incoming file. To be actually used, all selection criteria must match.

USER

User ID of sender.

Name or pattern of user ID of sender.

One of the selection criteria; incoming data set must have originated from a sender with a matching name for this receive entry to be further considered as a possible action for the incoming file. To be actually used, all selection criteria must match.

LOCAL USER

Name or pattern of local user ID who is the recipient of the incoming information.

You may specify something other than your own user ID only, if you are defined as privileged user

Default: your current user ID

To match a transfer from an MVS host, this field must be left blank or it must contain an asterisk (*) to match any ID, because rvsMVS sends to your station, not to an individual user.

One of the selection criteria; incoming data set must be intended for a local user with a matching name for this receive entry to be further considered as a possible action for the incoming file.

To be actually used, all selection criteria must match.

DSNAME

(Virtual) name of incoming data set.

One of the selection criteria; incoming data set must be known by a matching name for this receive entry to be further considered as a possible action for the incoming file.

To be actually used, all selection criteria must match.

DSNNEW

Specify name under which data set should be stored.

This is the only way in which you as a user can specify directory or library where the incoming file is to be placed. Be careful, however, if you used a pattern in any of the selection criteria, above; **DSNNEW** cannot contain any wild cards and more than one matching data set may arrive before you processed the old one.

If you do not specify a fully qualified data set name, rvs[®] will try to generate one,

using information from your current path or library.

The full name will be shown when the panel is redisplayed after the resident receive entry has been created. Please check that the name is what you intended it to be.

Make sure that (sub-)directories or libraries exist when files are being delivered. rvs[®] will not create directories or libraries and deliveries will fail, if they do not exist.

ACCOUNT

Account to which all rvs[®] activities relating to receiving and storing a data set is to be charged.

TIMESTAMP

Generate unique data set name when delivering file by adding a time stamp as dsn qualifier.

On systems where this is not possible, numerical values will be used as last qualifiers.

Choose :

- **Y** = yes
- **N** = no

Default: **N**

DISP

Determines what should be done with the data set when processing completes.

Choose :

- **K**(eep) Data set will be kept after processing (and cataloged, if these are separate actions on the local system).
- **D**(elete) Data set will be deleted after processing. This option only takes effect if there are jobs to process.

Default: **K**

REPLACE

Specifies what should be done, when a data set with the same name already exists.

Choose :

- **R**(eplace) This option is only meaningful, if **DISPOSITION = K** and a data set with the specified

	To be actually used, all selection criteria must match.
Remote station (id)	Name of remote station.
User ID of sender	One of the selection criteria; incoming data set must have originated from a sender with a matching name for this receive entry to be further considered as a possible action for the incoming file. To be actually used, all selection criteria must match.
Local user or alias	User ID of local recipient. One of the selection criteria; incoming data set must be intended for a local user with a matching name for this receive entry to be further considered as a possible action for the incoming file. To be actually used, all selection criteria must match.
Data set name	(Virtual) name under which the file is known during transfer. One of the selection criteria; incoming data set must be known by a matching name for this receive entry to be further considered as a possible action for the incoming file. To be actually used, all selection criteria must match.
New data set name	Name under which the data set will be stored in the local system.
Add timestamp	<ul style="list-style-type: none">• Y(es) Generate a unique data set name when delivering the file.• N(o) Otherwise
Disposition	<ul style="list-style-type: none">• K(eep) Data set will be kept after processing (and cataloged, if these are separate actions on the local system).• D(elete) Data set will be deleted after processing (e.g. in combination with print data).
Replace of data set	<ul style="list-style-type: none">• R(eplace) Replace existing data set• N(o) Create new data set name• I(gnore) Ignore incoming data set

Job to be started	Name of data set that will be started as a batch job when data is received.
Account	Account to which all rvs [®] activities relating to receiving and storing a data set is to be charged.
Comment	Brief comment describing the purpose of this entry (used for display, only).
Created by user	User ID of person who created this entry.
Creation date (date)	Date when this entry was created.
Creation date (time)	Time when this entry was created.
Last used date (date)	Date when this entry was last used to determine the actions that occurred when data was received.
Last used date (time)	Time when this entry was last used to determine the actions that occurred when data was received.

Confirm Delete Resident Receive Entry (mask CDRE)

In this mask, you are asked to confirm that you really want to delete this displayed resident receive entry.

Functionkey <F3> : Cancel delete request; the displayed entry will remain active.

Functionkey <F5> : The displayed entry will be deleted.

DATE Current date

TIME Current time

Remote station (id)	Station ID of the remote system One of the selection criteria; incoming data set must have originated at a station with a matching name for this receive entry to be further considered as a possible action for the incoming file. To be actually used, all selection criteria must match.
Remote station (name)	(Descriptive) name of the remote system.

User ID of sender	<p>One of the selection criteria; incoming data set must have originated from a sender with a matching name for this receive entry to be further considered as a possible action for the incoming file</p> <p>To be actually used, all selection criteria must match.</p>
Local user or alias	<p>User ID of local recipient.</p> <p>One of the selection criteria; incoming data set must be intended for a local user with a matching name for this receive entry to be further considered as a possible action for the incoming file.</p> <p>To be actually used, all selection criteria must match.</p>
Data set name	<p>(Virtual) name under which the file is known during transfer.</p> <p>One of the selection criteria; incoming data set must be known by a matching name for this receive entry to be further considered as a possible action for the incoming file.</p> <p>To be actually used, all selection criteria must match.</p>
Comment	<p>Brief comment describing the purpose of this entry (used for display only).</p>
Created by user	<p>User ID of person who has created this entry.</p>
Creation date (date)	<p>Date when this resident receive entry was created.</p>
Creation date (time)	<p>Time when this resident receive entry was created.</p>
Last used date (date)	<p>Date when this entry was last used to determine the actions that occurred when data was received.</p>
Last used date (time)	<p>Time when this entry was last used to determine the actions that occurred when data was received.</p>

Update Resident Receive Entry (mask REU)

The purpose of this panel is to update an existing resident receive entry. This mask is similar to the mask `Create Resident Receive Entry, CRRE`.

To update the displayed entry, overwrite text in those fields you want to change and confirm by pressing key **<ENTER>**. The old entry will be deleted and a new one will be created.

Functionkey **<F3>** : Cancel modification request; the displayed resident receive entry will remain unchanged.

Key **<ENTER>** : Update the existing resident receive entry.

STATION ID

Station ID of sender

To change this field, overwrite it with a new station ID; or blank it out to get a list of all available station-ids to select a new one.

This is one of the selection criteria; incoming data set must have originated at a station with a matching name for this receive entry to be further considered as a possible action for the incoming file.

To be actually used, all selection criteria must match.

USER

User ID of sender.

One of the selection criteria; incoming data set must have originated from a sender with a matching name for this receive entry to be further considered as a possible action for the incoming file.

To be actually used, all selection criteria must match.

LOCAL USER

Name or pattern of local user ID who is the recipient of the incoming information.

You may specify something other than your own user ID, only if you are defined as privileged user.

Default: your current user ID

This is one of the selection criteria; incoming data set must be intended for a local user with a matching name for this

receive entry to be further considered as a possible action for the incoming file.

To be actually used, all selection criteria must match.

DSNAME

Name of incoming data set

(Virtual) name under which the file is known during transfer.

One of the selection criteria; incoming data set must be known by a matching name for this receive entry to be further considered as a possible action for the incoming file.

To be actually used, all selection criteria must match.

DSNNEW

New name of data set that should be given to the recipient's copy of the received data.

ACCOUNT

Account in which all rvs[®] activities relating to receiving and storing a data set is to be charged.

TIMESTAMP

Data set name should be made unique by adding of time stamp.

Choose :

- **Y** = yes
- **N** = no

Default: **N**

DISP

Determines what should be done with the data set when processing completes.

Choose :

- **K**(eep) Data set will be kept after processing (and cataloged, if these are separate actions on the local system).
- **D**(elete) Data set will be deleted after processing. This option only takes effect if there are jobs to process.

Default: **K**

REPLACE

Specifies what should be done, when a data set with the same name already exists.

Choose :

- **R**(eplace) This option only takes effect,

if **DISPOSITION = K** and a data set with the specified name does already exist.

- **N(ew)** Create a new data set name.
- **I(gnore)** Incoming data set will be ignored.

Default: **N**

JOB

Name of data set that can be started as a batch job, to specify a following process.

COMMENT

Brief comment describing the purpose of this entry (used for display only).

5.5.2 Job Start after Send Attempt

This chapter contains a description of how to operate with job starts after send attempt entries.

Note: The behaviour of rvs when a Job Start after Send Attempt is carried out is influenced by the rvs parameter **JSERRHOLD**. Please refer to chapter 8.1.

Job Start after Send Attempt Entries (mask JS)

```
|-----|
| R V S - LIST / UPDATE / DELETE OF SEND-JOB-ENTRIES |
|-----|

specify parameter of send-job-entries:

1. STATION-ID      ===>                station id of receiver
2. USER-ID        ===> *                user id of sender
3. DATASETNAME     ===> *                virtual datasetname
4. SENDATTEMPTS   ===> *                number of sendattempts

F1 = HELP, F3/ESC = CANCEL, EINGABE = show list of entries
```

The purpose of this panel is to select criteria, you want to access. Specify the following parameters to describe the job start after send attempt entry you want to access.

Functionkey **<F1>** : Display help information.

Functionkey **<F3>** : Cancel request; all receive entries remain unchanged.

SID	Station ID of receiver If this field is blank, a list of available station IDs will be displayed. Use key <S> to select a station ID from that list and confirm your choice by pressing <ENTER> . Your choice will be inserted into the current panel.
USER	User ID of sender Name or pattern of user ID of sender. One of the selection criteria; outgoing data set must have originated from a user with a matching name for this JobStart entry to be further considered as a possible action for the outgoing file. To be actually used, all selection criteria must match.
DSN	Name of data set to send Enter name of data set of Job Start after Send Attempt entry. Leave blank or specify pattern to get a data set selection list.
ATTEMPTS	Number of (failed) send attempts before starting a job.

List of Job Start after Send Attempt Entries (mask JSSL)

This panel displays a list of Job Start after Send Attempt Entries.

Use line command **<S>**, **<U>**, or **<D>** to get more information about the entry, update, or delete it, respectively.

Functionkey <F1> :	Display help information.
Functionkey <F2> :	Display a mask to add a new entry.
Functionkey <F3> :	Ignore any selection and leave display.
Functionkey <F7> :	Page up.
Functionkey <F8> :	Page down.
Key <ENTER> :	Confirm selection.

DATE Current date

TIME Current time

Specify option Enter one of these options to the left of one of the displayed JobStart after SendAttempt entries:

- **S**: Display the entry.
- **U**: Update the entry.
- **D**: Delete the entry.

SID Station ID of receiver.

Data set name Name or name pattern of outgoing data set that will be processed by this Job Start entry.

Job Name Job that will be started after the specified amount of send attempts.

Attempts Number of send attempts before the specified job will be started.

Create Job Start after Send Attempt Entry (mask CJSS)

```
-----  
| R V S - CREATE SEND-JOB-ENTRY |  
-----  
Specify, which program has to start after sending of a specific dataset  
(Attempts=0) or if it was not possible to send the dataset in a specific  
number of attempts (Attempts>0) :  
  
1. Station ID      ==> |          station id of receiver  
2. User ID        ==> *          user ID of sender  
3. virt. datasetname ==> *          virtual datasetname  
4. Attempts       ==> 0    number of send-attempts  
5. Program        ==>          Program to start  
6. Comment        ==>  
  
F1 = HELP, F3/ESC = CANCEL, ENTER = CREATE ENTRY
```

In this panel, you can create job start after send attempt entry.

In a job start after send attempt entry, a user may specify what actions should be taken, when a data set has been attempt to send.

Functionkey <F1> : Display help information.

Functionkey <F3> : No entry will be created.

Key <ENTER> : If all fields are specified the entry will be created.

Station ID

Station ID of receiver.

If this field is blank, a list of available station IDs will be displayed. Use key <S> to select a station ID from that list and confirm your choice by pressing <ENTER>.

Your choice will be inserted into the current panel.

This is one of the selection criteria; outgoing data set must have destined to a station with a matching name for this entry to be further considered as a possible action for the outgoing file. To be actually used, all selection criteria must match.

User ID

User ID of sender.

Name or pattern of user ID of sender.

One of the selection criteria; outgoing data set must have originated from a sender with a matching name for this entry to be further considered as a possible action for the outgoing file.

To be actually used, all selection criteria must match.

Attention: The user ID as selection criteria is not yet supported.

virt. datasetname

(Virtual) name of outgoing data set.

One of the selection criteria; outgoing data set must be known by a matching name for this entry to be further considered as a possible action for the outgoing file.

To be actually used, all selection criteria must match.

Attempts

One of the selection criteria; specify

amount of send-attempts after which the specified job should be started.

If **Attempts** is greater than 0, the job will be started after the specified amount of failed send attempts.

If **Attempts** is 0, the job will be started after a successful transmission.

Default: **0**

Program Name of data set that can be started as a batch job, to specify the following process.

Comment Brief comment describing the purpose of this entry (used for display only).

Display Job Start after Send Attempt Entry (mask SJSS)

```
|-----|
|          R V S - SEND-JOB-ENTRY DETAILS          |
|-----|
|                                                     |
|                                                     |
|                                                     |
| Command number      : 341                          |
| remote station      : SUM                          |
| User-ID sender      : *                            |
| virtual datasetname : LOOPTEST                     |
| Program to start    : /home/umr/bin/lisa           |
| Job will be started in case of successful transmission |
| :                                                         |
| Comment             :                               |
| last used at        :                               |
| created by          : umr                            |
|                                                     |
|                                                     |
| F1 = HELP, F3/ESC = EXIT                           |
|-----|
```

In this panel, displays details about the Job Start after Send Attempt entry you selected in the previous panel.

Functionkey **<F1>** : Display help information.

Functionkey **<F3>** : Leave display and return to previous panel.

DATE Current date

TIME Current time

CmdndNbr. Internal number of this entry.

Station ID Station ID of receiver.

If this field is blank, a list of available station IDs will be displayed. Use key **<S>** to select a station ID from that list and confirm your choice by pressing **<ENTER>**.

Your choice will be inserted into the current panel.

This is one of the selection criteria; outgoing data set must have destined to a station with a matching name for this entry to be further considered as a possible action for the outgoing file.

To be actually used, all selection criteria must match.

User ID sender

User ID of sender.

Name or pattern of user ID of sender.

One of the selection criteria; outgoing data set must have originated from a sender with a matching name for this entry to be further considered as a possible action for the outgoing file.

To be actually used, all selection criteria must match.

Attention: The user ID as selection criteria is not yet supported.

virtual DS-Name

(Virtual) name of outgoing data set.

One of the selection criteria; outgoing data set must be known by a matching name for this entry to be further considered as a possible action for the outgoing file.

To be actually used, all selection criteria must match.

Program to start

Name of data set that can be started as a batch job, to specify the following process.

Job will be started...

One of the selection criteria; specifies amount of send-attempts after which the specified job should be started.

Comment

Brief comment describing the purpose of this entry (used for display only).

Last used at

Date and time when this entry was last used to determine the actions that occurred when a data set was attempt to send.

created by User ID of person who created this entry.

Confirm Delete of Job Start after Send Attempt Entry (mask DJSS)

In this panel, you are asked to confirm that you really want to delete this displayed Job Start after Send Attempt entry you selected in the previous panel.

Functionkey <F1> : Displays help information.

Functionkey <F3> : Cancels delete request; the displayed entry will remain active. Leave display and return to previous panel.

Functionkey <F5> : Confirms delete request; the displayed entry will be deleted

DATE Current date

TIME Current time

CmndNbr. Internal number of this entry.

Remote Station Station ID of receiver.

User ID of sender User ID of sender.

Name or pattern of user ID of sender.

virtual data setname (Virtual) name of outgoing data set.

Attempts Specifies amount of send-attempts after which the specified job should be started.

Job Name of data set that can be started as a batch job, to specify the following process.

Comment Brief comment describing the purpose of this entry (used for display only).

Created by user User ID of person who created this entry.

Creation date Date and time when this entry was created.

Last used date Date and time when this entry was last used to determine the actions that occurred when a data set was attempt to send.

Update Job Start after Send Attempt Entry (mask UJSS)

In this panel, you can update Job Start after Send Attempt entry. In a Job Start after Send Attempt entry, a user may specify what actions should be taken, when a data set has been attempted to send. This mask is similar to the mask `Create Job Start after Send Attempt Entry, CJSS`.

Functionkey <F1> : Displays help information.

Functionkey <F3> : Terminates action and return to previous panel.

Key <ENTER> : If all fields are specified the entry will be updated.

STATION ID	Station ID of receiver. This is one of the selection criteria and so you will not be able to change this entry.
USER	User ID of sender. This is one of the selection criteria and so you will not be able to change this entry.
DSNAME	(Virtual) name of outgoing data set. This is one of the selection criteria and so you will not be able to change this entry.
ATTEMPTS	Number of send-attempts before starting the specified job. This is one of the selection criteria and so you will not be able to change this entry.
JOB	Name of data set that can be started as a batch job, to specify the following process..
COMMENT	Brief comment describing the purpose of this entry (used for display only).

5.5.3 User List

This chapter describes how you can create, display, confirm and update user entries.

List of User Entries (mask USRL)

```
-----
|          R V S  -  DISPLAY / UPDATE / DELETE USER
-----
                                Date = 2000/03/29 Time = 09:58

  Optionen: S - display   U - update   D - delete

User-ID      full Name                Authorization   Language
-----
| bwa
pw           ?                        Administrator  Deutsch
-----

-----
  F1 = HELP, F2 = NEW ENTRY, F3/ESC = EXIT, F5 = REFRESH
Row 1 .. 2 - End of table reached
-----
```

This panel displays a list of rvs[®] user entries.

Use line command **<S>**, **<U>**, or **<D>** to get more information about the entry, update, or delete it, respectively.

- Functionkey **<F1>** : Display help information.
- Functionkey **<F2>** : Display a mask to add a new entry.
- Functionkey **<F3>** : Ignore any selection and leave display.
- Functionkey **<F7>** : Page up.
- Functionkey **<F8>** : Page down.
- Key **<ENTER>** : Confirm selection.

DATE Current date
TIME Current time

Specify option Enter one of these options to the left of one of the displayed user entries:

- **S**: Display the entry.
- **U**: Update the entry.
- **D**: Delete the entry.

UID User ID.
full Name full Name or description of the user.

Authorization	Authorization to use rvs [®] .
Language	Language to display panels and messages for this user.

Create User Entry (mask CUSR)

```
-----  
|                R V S  -  CREATE USER-ENTRY                |  
-----
```

```
1. User-ID          : | ██████████  
2. full Name       :  
3. Authorization   : U                A=Admin  O=Operator  U=User  
4. Language        : D                D=Deutsch  E=English
```

F1 = HELP, F3/ESC = CANCEL, RETURN = CREATE ENTRY

In this panel you can create a rvs[®] User Entry.

In a user entry you can grant authorizations to work with rvs[®]. Furthermore, you can specify the language to display masks and messages for the specified user.

- Functionkey <F1> : Display help information.
- Functionkey <F3> : No entry will be created; leave the display and return to the previous one.
- Key <ENTER> : If all necessary fields are filled the entry will be created.

User ID	User ID which identifies the user on the system.
full Name	full Name or description of the user.
Authorization	Authorization to use rvs [®] . This authorization has no affect on the system authorization of this user. <ul style="list-style-type: none">• A - Administrator• O - Operator• U - User

Language Language to display panels and messages for this user.

- **D** - German
- **E** - English

Password Password for this user.

Display User Entry (mask SUSR)

This panel displays details about the rvs[®] User Entry you selected in the previous panel.

Functionkey <**F1**> : Display help information.

Functionkey <**F3**> : leave the display and return to the previous one.

DATE Current date.

TIME Current time.

User ID User ID which identifies the user on the system.

full Name full Name or description of the user.

Authorization Authorization to use rvs.

This authorization has no affect on the system-authorization of this user.

- **A** - Administrator
- **O** - Operator
- **U** - User

Language Language to display panels and messages for this user.

- **D** - German
- **E** - English

Password Password for this user.

Confirm Delete of User Entry (mask DUSR)

In this panel, you are asked to confirm that you really want to delete the displayed user entry.

- Functionkey <F1> : Display help information.
- Functionkey <F3> : Cancel delete request; the displayed entry will remain active.
Leave display and return to previous panel.
- Functionkey <F5> : Confirm delete request; the displayed entry will be deleted

DATE Current date.

TIME Current time.

User ID User ID which identifies the user on the system.

full Name full Name or description of the user.

Authorization Authorization to use rvs®.

Language Language to display panels and messages for this user.

Password Password for this user.

Update User Entry (mask UUSR)

In this panel you can update a rvs® User Entry. This mask is similar to the mask Create User Entry, CUSR.

In a user entry you can grant authorizations to work with rvs®. Furthermore, you can specify the language to display masks and messages for the specified user.

- Functionkey <F1> : Display help information.
- Functionkey <F3> : No entry will be created; leave display the display and return to the previous one.
- Key <ENTER> : If all necessary fields are filled the entry will be updated.

User ID	User ID which identifies the user on the system.
full Name	full Name or description of the user.
Authorization	Authorization to use rvs [®] . This authorization has no affect on the system-authorization of this user. <ul style="list-style-type: none">• A - Administrator• O - Operator• U - User
Language	Language to display panels and messages for this user. <ul style="list-style-type: none">• D - German• E – English
Password	Password for this user.

5.5.4 Information about rvs[®]

Information (mask INFO)

OPTION	Select one of these options: <ol style="list-style-type: none">1. RVS NEWS (if available) may be listed, using this option (not yet implemented).2. General Informations about rvs[®] may be listed (not yet implemented).3. Lists all nodes that can be reached from the local station (either directly or indirectly).4. Print rvs[®] User Manual (not yet implemented). Information for options 1 and 2 is provided and maintained by your local rvs [®] administrator.
copies	Specify how many copies you want to print of rvs [®] User Manual

6 Database Maintenance

The rvs[®] Database is the memory of rvs[®], both as to what happened in the past and what still has to be done. If the rvs[®] Database is damaged or contains inconsistent entries, files may be sent twice or not at all.

rvsX 2.05 and above gives the possibility of binding to an Oracle database. The rvs[®] internal C-ISAM database is replaced with the external high performance Oracle database. Please read the chapter 11 "rvsX Oracle Binding (rvsX High Performance)" for more information.

This chapter describes how to maintain the rvs[®] Database for rvsX. The maintenance of the Oracle database and the C-ISAM database is the same.

6.1 Backup

Backup and recovery described in this section can only be used if you have set the parameter **DBLOG=Yes** in the installation process. If you choose **DBLOG=No**, you are not able to use this feature!

Regular backup of all database related files is very important. How often you should do it depends upon your usage of rvs[®]; if you are an intensive user, LOG files may become pretty large if the interval between backups is too long. The larger the LOG files the more time consuming a database recovery will be.

To create a backup

- Choose a time with no or little rvs[®] traffic.
- Make sure that no rvs[®] transmissions are active. Simplest method is to disable the rvs[®] related connections in TRANSIT. This is also necessary in order to prevent receiver tasks to be started from remote. Running transmissions would update the database which could be harmful during backup. Because of the rvs[®] restart facility, the interrupted transmissions will be restarted automatically at the point of rupture when normal processing resumes.
- Edit your environment data set and add or change the **MODE** statement **MODE=d**. This will disable access to the database by programs, procedures or dialog interface.
- Delete obsolete database entries (rvs[®] operator command `cleanup days=n ss=yes` with 'n' days being the maximum age of kept entries).
- Stop Monitor (rvs[®] operator command `stop`).
- Use your favorite method to create a backup (or copy) of all files (see chapter "Representation means" for the detailed description of **\$RVSPATH**)
 - `$RVSPATH/db/*.DB`,
 - `$RVSPATH/db/*.IDX`, and

- \$RVSPATH/ rvs/db/rldb.log.
- Delete file \$RVSPATH/db/rldb.log
- Edit environment data set, change **MODE** statement to **MODE=n**.
- Restart communications manager
- Restart rvs[®] Monitor.

6.2 Recovery

Backup and recovery described in this section can only be used if you have set the parameter **DBLOG=Yes** in the installation process. If you choose **DBLOG=No**, you are not able to use this feature!

If your database should get damaged the database can be recovered from the latest backup. For forward recovery from latest backup the database log file, that has been written since the last backup and which contains all the database changes since then, will be used as input.

- Disable all rvs[®] related connections in TRANSIT. This is necessary in order to prevent receiver tasks to be started from remote and to end running transmissions. Both would update the database which could be harmful during backup. Because of the rvs[®] restart facility, the interrupted transmissions will be restarted at the point of rupture when normal processing resumes.
- Edit your environment data set and add or change the **MODE** statement **MODE=d**. This will disable access to the database by programs, procedures or dialog interface.
- Delete obsolete database entries (rvs[®] operator command `cleanup days=n ss=yes` with 'n' days being the maximum age of kept entries).
- Stop Monitor (rvs[®] operator command `stop`).
- Use your latest backup version to restore all files
 - - \$RVSPATH/db/*.DB and
 - - \$RVSPATH/db/*.IDX .(see chapter "Representation means" for the detailed description of **\$RVSPATH**).
- Note: Do not restore LOG file (\$RVSPATH/db/rldb.log).
- Rename database log \$RVSPATH/db/rldb.log to \$RVSPATH/db/rldb.old.log (using **UNIX** command `ren $RVSPATH/db/rldb.log rldb.old.log`).
- Start Monitor for database recovery ('Wiederanlauf'):
`rvsmon /w/$RVSPATH/db/rldb.old.log`
- Check LOG file \$RVSPATH/db/rlog.log for any warning or error messages.
- After successful recovery, create a backup of the rvs[®] database (see above).
- Edit environment data set, change **MODE** statement to **MODE=n**.
- Restart communications manager.
- Restart Monitor.

6.3 Dump rvs[®] Database (rvsddb)

rvsddb creates the textfile `rldbdbdump.log` in the `arcdir` directory containing the information from the rvs[®] database.

```
rvsddb [/e<envdsn>] [/l<x>]
```

where

- the optional parameter `/e` is used only, if the environment data set is not defined in the **RVSENV** environment variable and not located in the current directory, either.
- the optional parameter `/l` defines the language (**x**) to be used for prompts and messages.

Stop the rvs[®] system before executing `rvsddb`.

For each entry in the rvs[®] database `rvsddb` creates one record in the textfile, containing tablename columnnames and values.

6.4 Dump rvs[®] User and Job Starts (rvsdru)

rvsdru creates the textfile `rlrudump.log` in the `$RVSPATH/arcdir` directory of rvs[®] (see chapter "Representation means" for the detailed description of **\$RVSPATH**).

It contains the informations about rvs[®] users, resident receive entries and job starts after send attempts stored at the rvs[®] database.

```
rvsdru [/e<envdsn>] [/l<language>]
```

where

the optional parameter `/e` is used only, if the environment data set is not defined in the **RVSENV** environment variable and not located in the current directory, either.

The textfile `rlrudump.log` can be used as an input file to the rvs[®] batch interface (`rvsbat`), i.e. a new database is to create and the user, resident receive entries and job starts after send attempts should be taken from the old database.

```
rvsbat /i/$RVSPATH/arcdir/rlrudump.log
```

To get further informations about the batch interface of rvs[®] see "Reference Manual".

The shell script `s_rvsbackup` gives a sample how to use

- `rvsddb`
- `rvsidb`
- `rvswdb`

in order to create a backup of rvs[®] and cleanup the rvs[®] database.

6.5 Delete rvs[®] Database (`rvsdbdel`)

`rvsdbdel` lets you delete your old database and (optionally) remove all temporary files.

```
rvsdbdel [/e<envdsn>] [/l<x>] [-?] [-d[dumpfilename]] [-f]
```

where

- the optional parameter **/e** is used only, if the environment data set is not defined in the **RVSENV** environment variable and not located in the current directory, either.
- the optional parameter **/l** defines the language (**x**) to be used for prompts and messages.
- with the optional parameter **-d** you can define the dumpfile to which the old database can be stored.
- with the optional parameter **-f** you can delete the database without any inquiry.

Stop the rvs[®] system before executing `rvsdbdel`.

6.6 Create new Database (`rvsidb`)

If your database got damaged, it is necessary to delete and recreate it. The following steps are needed:

- Stop rvs[®] system
- Delete old database:
`rvsdbdel [/e<envdsn>] [/l<x>] [-?] [-d[dumpfilename]] [-f]`
where
 - the optional parameter **/e** is used only, if the environment data set is not defined in the **RVSENV** environment variable and not located in the current directory, either.
 - the optional parameter **/l** defines the language (**x**) to be used for prompts and messages.
 - with the optional parameter **-d** you can define the dumpfile to which the old database can be stored.
 - with the optional parameter **-f** you can delete the database without any inquiry.
- Create new database:
`rvsidb [/e<rvsenv>] [/i<cmdfile>] [/l<language>] lid`
 - the optional parameter **/e** is used only, if you do not generate the default database. If you generate the database in another directory, the path information is read from the environment data set specified as **rvsenv**.
Note, that there is no blank allowed between **/e** and **rvsenv**.
 - **lid** is replaced by your local station ID which can consist of up to 16 characters.
 - the optional parameter **/i<cmdfile>** can be used to read another input file instead of `rdstat.dat`: **file** can be a single input file or a directory which contains several input files.

- the optional parameter **/l<language>** can be used to change the default parameter **e** (english language) to german language (**d**).

Note, that all incomplete transmissions are irretrievably lost, of course, when you delete the old rvs[®] database.

6.7 Write rvs[®] Database (rvswdb)

rvswdb reads the textfile rldbdump.log in the arcdir directory containing information from the rvs[®] database and store this Information in the rvs[®] database.

```
rvswdb [/e<envdsn>] [/l<x>] /i<inputfile>
```

where

- the optional parameter **/e** is used only, if the environment data set is not defined in the **RVSENV** environment variable and not located in the current directory, either.
- the optional parameter **/l** defines the language (**x**) to be used for prompts and messages.
- the mandatory parameter **/i** defines the inputfile, from which the data is read, rldbdump.log from arcdir directory.

Stop the rvs[®] system before executing rvswdb.

For each entry in the textfile, rvswdb searches the rvs[®] database and create or update an entry by its primary key.

6.8 Cleanup

Searching in a large database generally takes longer than looking for something in a small one; a larger database, however, retains more information on completed transmissions.

The relative importance of these two goals for your installation determines, how often you will want to clean up the rvs[®] database and for how long you want to keep information about ended or deleted transmissions. Use

```
cleanup days=n ss=yes
```

from the console prompt; or include it among the Monitor's initial commands in \$RVSPATH/init/rdmini.dat (see chapter "Representation means" for the detailed description of \$RVSPATH) so that it will be executed, whenever the rvs[®] Monitor is started. **n** specifies the retention period in days (in 24 hour periods, actually); **ss=yes** deletes all SendStatistic records.

7 Operator Console and Commands

This section gives an overview of the operator commands and how to use them.

Operator Console

The Operator Console is used to control the functions of rvs[®]. The following section presents the rvs[®] log book and possibilities for control and parameter setting in rvs[®] by different commands.

The Operator Console displays all messages created since the last start of the rvs[®] Monitor. The entire log book is generated in the following file:

- `$RVSPATH/db/rlog.log`

and can be saved in an archive file for documentation purposes (see chapter "Representation means" for the detailed description of **\$RVSPATH**).

To **start** the Operator Console:

- write `$RVSPATH/system/rvscns`

To **terminate** the Operator Console:

- press the `esc` button

To **scroll** in the Operator Console:

- press `<` or `>`

To **enter commands** in the Operator Console:

- you are automatically in the command mode; confirm your command by `[ENTER]`

Operator Commands

Most operator commands provide information about or influence contents of items in the area of

- individual rvs[®] commands
- traffic to neighbors
- stationtable
- database
- rvs[®] parameters

Command syntax is specified in chapter 7.6 "Patterns". The sections in this chapter provide a brief overview of the functionality of operator commands.

Note the difference between operator commands and rvs[®] commands:

operator command is a command like `listcmd`, `modst`, or `cleanup` that you enter

- in the Operator Console

to instigate some action by the rvs[®] Monitor.

rvs command is a database entry (like `SE` (send entry), `SK` (send command), or `BB` (user notification)). These commands, representing a processable unit of work, are created internally by rvs[®] and are used to organize the flow of work and information between its components.

7.1 Work with Individual rvs[®] Commands

When a user creates a send entry (rvs[®] command `SE`), the user can hold it, delete it, or release it in the dialog interface `rvsdia` as long as the Monitor did not start processing it. Once one or more send commands (rvs[®] command `SK`) have been created, command execution can only be modified through operator commands.

7.1.1 List information about one command

In the Operator Console use `listcmd cn=n` to display information about individual commands. 'n' is the command number which uniquely identifies every rvs[®] command. Command numbers are displayed at the Operator Console when the command is created and when certain actions are carried out.

```
NEW_CMD_CREATED IZ(245) created from IE(242)
```

tells you, for example, that an information delivery command (`IZ`) with command number 245 has been created from information entry (`IE`) with command number 242 (which in turn normally will have been created by the Receiver while accepting a data set from another rvs[®] node). To learn more about the above information delivery, enter

```
listcmd cn=245
```

If this operator command is entered after `IZ` terminated, the system response may look like this:

```
R:    KT(245) :
R:          CMDTYPE      = IZ
R:          PRIORITY     = 70
R:          DTCREATED    = 2002/03/25 15:12:29
R:          DTSCHEDULE   = 2002/03/25 15:12:29
R:          DTDONE       = 2002/03/25 15:12:43
```

```
R:          FLAGSERIAL      = FALSE
R:          STATUS          = en
R:          ERRORCODE       = 0
R:  IZ (245) :
R:          CNIE            = 242
R:          DSNLOCAL        = $RVSPATH/usrdat/AUTOSER1
```

KT (the part of the information that is stored in the command table) contains the same fields for all commands:

CMDTYPE	command type, in this case IZ (information delivery, from German InformationsZustellung)
PRIORITY	the priority with which the command is or was executed; the smaller this number, the higher the priority. These priorities are defined as rvs [®] parameters BBPRIO , IEPRIO , IZPRIO , etc.
DTCREATED	date and time when the command was created
DTSCHEDULE	date and time when the command will be available for execution (or in case of an ended command was scheduled to be executed). Very often, DTCREATED and DTSCHEDULE will be the same; command retry or user actions may be the reasons behind differing values.
DTDONE	date and time when processing of this command terminated.
FLAGSERIAL	indicates, whether the (SE) command (still) is in serialization hold; i.e. it is waiting for another rvs [®] command to terminate before it will be eligible for processing.
STATUS	current status; see description of listcmd in chapter 7.1.2 "List Command Numbers" for a list of first status letters. The second letter tells you, whether it is a `n'ormal or a `t'est command. _ In our example, `en' means that the `n'ormal command `e'nded.
ERRORCODE	tells you whether the previous attempt to execute the command was successful or failed. In particular, if the status is `q'ueued and the ERRORCODE is not zero, error recovery will be performed, when the

command will be selected for execution.

DSNLOCAL

The detailed description of **\$RVSPATH** is explained in chapter "Representation means".

This general information is followed by command-specific information. In our example, it tells you that the parent command (the **IE**) had command number 242 and in which file the received information was stored.

7.1.2 List Command Numbers

To get a list of all commands that are in a certain status, use `listcmd` with parameter status.

```
listcmd status=q
```

will give you a list of all `q'ueued commands, for example. See description of `listcmd` in chapter 7.7 "Command Descriptions" for a list of status letters. The second part of the status (`n'ormal or `t'est) is always taken from your current execution mode and you will only see commands that belong to this mode.

7.1.3 Hold, Free or Delete an rvs[®] command

Use

`holdcmd cn=n` to place command with command number `n` in operator hold, so that it will not be executed until explicitly freed;

`freecmd cn=n` to free command with command number `n` previously placed in operator or user hold; and

`delcmd cn=n` to delete command with command number `n`.

If you want to delete a command, first you have to put it into the hold state. You will have to delete all commands, too, which are generated from the send entry (i.e. **SE** and **SK**). Otherwise, the database would be left in an inconsistent state. This applies to file reception, too (i.e. **IE** and **IZ** should be deleted).

Note, that no prompting for confirmation has been implemented, yet! So, be careful that you do not mistype the command number when deleting a command.

7.2 Suspend Traffic to Neighbor

`holdcmd` and `freecmd` may also be used to suspend all traffic to a particular neighboring node in the rvs[®] network. When you know that communication lines to a neighbor will not be available for a while, suspending all traffic to that node will prevent unnecessary send attempts and is much easier than holding and freeing

all queued commands for that neighbor, individually. Also, all newly created sends to that station will automatically be suspended.

```
holdcmd sid=xxx
```

suspends all traffic to station ID `xxx' (which must be a neighbor), while

```
freecmd sid=xxx
```

lifts the suspension and releases all suspended commands (commands that were placed in hold using holdcmd cn=n are not released).

Use `listst sid=xxx` and check flag **FLAGSUSPND** in **NK**: listing to find out whether all traffic to `xxx' is suspended (**FLAGSUSPND=TRUE**) or not (**FLAGSUSPND=FALSE**).

7.3 Activate Neighbor

activate can be used to test the connection or to get queued files from your partner.

```
activate sid=xxx  
act sid=xxx
```

activate starts a send process `rvscom` which establishes a connection to the neighbor station. Your neighbor is a partner station to which you have a direct connection (by TCP/IP, LU 6.2 or X.25/ISDN). If your rvs[®] station has queued files for this partner, they will be send now. If your partner has queued files, they will be received. Otherwise, the connection ends.

The partner can be called repeatedly, for example every 3 hours:

```
opcmd cmd="act sid=xyz" repeat=03:00:00
```

Note, that you can't activate the target station, if you are sending a file via routing.

7.4 Work with Stationtable Entries

Entries in stationtable **ST** and its related tables

- LU** LU 6.2 (or APPC) parameters,
- TC** TCP/IP parameter table,
- NK** neighboring nodes (from German `NachbarKnoten'),
- OP** ODETTE parameters, and
- RT** routing table
- XP** X.25 native and ISDN parameters

are explained in chapter 3 "Configuration of rvsX". Use

```
listst sid=xxx
```

to list all entries in these tables relating to station ID `xxx`.

To modify entries, edit `RVSPATH/init/rdstat.dat` (see chapter "Representation means" for the detailed description of **RVSPATH**), the file that provides stationable definitions when the database is generated by `rvsidb`. If you are using the default file `RVSPATH/init/rdstat.dat`, use

```
modst
```

without parameters. Else create a separate file in directory `RVSPATH/init/` which contains only new and modified entries and set the filename parameter **DSN** (DataSetName) to the path name of your file.

```
modst dsn="$RVSPATH/init/rdstat_new.dat"
```

reads file `RVSPATH/init/rdstat_new.dat` and replaces all database table rows with the input of this file. Therefore, specify all fields for each entry that you want to replace! The file name can be a single input file or a directory which contains several input files.

```
delst sid=xxx
```

removes all entries relating to station ID `xxx` from the stationtable and its related tables from the database (it does not search for open send requests to this station, however; if there are any, they will fail later on, when Monitor or Sender will try to execute them).

We recommend that you execute the command

```
delst sid=xxx
```

first and then `modst`, to avoid possible duplicate entries in the database. Note, that no prompting for confirmation has been implemented, yet! So, be careful that you do not mistype the station ID when deleting.

7.5 Work With **rvs**[®] Parameters

Parameters can be used to customize **rvs**[®] operations. They are described in chapter "**rvs**[®] Parameters"; tips how to choose **rvs**[®] parameter values are given in chapter "**rvs**[®] Parameter Values".

Use

```
listparm name
```

to list one or more parameter values. **NAME** can be

- the name of a parameter to list this one value,
- a pattern to list all parameters whose names match this pattern (patterns are described in section 7.6 "Patterns"), or

- **ALL** to list all parameters.

Parameter **NAME** can be modified with

```
setparm name=value
```

No plausibility check is made on the value you specify and results are unpredictable if you specify invalid values or wrong data types.

7.6 Patterns

Some commands support patterns, which means that the values you specify for these parameters may include wildcards asterisk (*) and/or question mark (?) to select more than one value at a time:

- * matches any number of arbitrary characters,
- ? matches exactly one arbitrary character

pattern must be enclosed in single or double quotation marks. For example, to list the execution priorities of all commands, enter

```
listparm "*prio"
```

and the Monitor will list the values of **BBPRIO**, **IEPRIO**, **IZPRIO**, etc. Specifying

```
listparm "q?prio"
```

will list **QEPRIO** and **QZPRIO**.

7.7 Command Descriptions

In the command descriptions, brackets ([]) indicate optional parameters, braces ({ | }) list alternatives.

```
activate SID=sid
```

act is an alias of **activate**. Activate a partner station. A rvs[®] communication program starts and establishes the connection. Queued data sets will be transmitted.

```
cleanup [DAYS=n] [SS=YES]
```

Physically delete commands from rvs[®] database that ended at least n days ago (**n** x 24 hours, to be precise). **cleanup DAYS=0** purges all ended and deleted commands. If **SS=YES** is specified, all old SendStatistics records are deleted.

```
delcmd CN=cn
```

Delete command with command number **cn** logically from database

```
delst SID=sid
```


Delete station **sid** from all station-table related database tables.

`freecmd` [CN=cn] [SID=sid]

Free command with command number **cn** from hold status or free all suspended transfers to neighbor **sid**.

`holdcmd` [CN=cn] [SID=sid]

Put command with command number **cn** in hold status or suspend all transfers to neighbor **sid**.

`listdbv` list version and creation date of database.

`listcmd` [CN=n] [STATUS=x]

`lc` is an alias of `listcmd`.

List details of command with number **n** or type and number of all commands whose status is **x**:

- a** active: the command is being processed
- d** deleted: this entry was (logically) deleted
- e** ended: processing of the command ended
- f** forwardable: ready to be sent (SK or QS)
- h** held: command was put in hold status; it cannot be processed until freed (i.e. put into status **q**)
- i** in transit: command is currently being sent (SK or QS)
- p** pending: processing partially complete; command is waiting for some event (e.g. SE is in this status, after SKs have been created until all receipts are received; RE is always in this status, waiting for information that matches it to come in)
- q** queued: ready to be processed
- s** suspended: ready to be sent but suspended, because all traffic to neighbor has been stopped (SK or QS)

`listparm` {name | "pattern" | ALL}

`lp` is an alias of `listparm`

List one, more, or all rvs[®] parameters.

`listst` SID=sid

`ls` is an alias of `listst`

list all stationtable entries involving station ID **sid**.

`modst` DSN=dsn

Modify stationtable ST and/or related tables by

applying commands stored in **DSN** which must be fully qualified.

Note: If you have made changes in the station table, afterwards you have to use the `modst` command in the operator console (`rvscns`). Only in this way the changes will take effect on the `rvs`[®] database.

`modst` can also be used without parameters; then the default `rdstat.dat` in the `init` directory is used as input.

The parameter **DSN** (data set name) can be used to read another input file instead of `rdstat.dat`: the file name can be a single input file or a directory which contains several input files (see chapter "Representation means" for the detailed description of **\$RVSPATH**):

```
modst DSN="$RVSPATH/init/otherfile.dat"
```

Default:

- `$RVSPATH/init/rdstat.dat` for **UNIX**

```
opcmd      [DSN=dsn] [CMD=cmd]
           [TIME=hh[:mm[:ss]]]
           [REPEAT=hh[:mm[:ss]]]
```

read operator commands from external data set ``dsn'` which must be fully qualified; or execute operator command specified in `cmd`; if **TIME** is specified, the operation will be rescheduled for the specified time of day; if **REPEAT** is specified, the command will be executed immediately and repeated (indefinitely) after the given interval has expired.

```
setparm    name=value
```

```
sp         is an alias of setparm
```

Modify one `rvs`[®] parameter.

```
start     [XMT] [CID=console-id]
```

Start MasterTransmitter or one console.

```
stop      [XMT] | [RVS={END|FORCE}] | XMT=END |
          CID={console-id|pattern}]
```

Stop Monitor (normally or with **FORCEDEND**), MasterTransmitter, or one or more consoles.

`stop xmt` and `stop XMT=END` are equivalent commands.

`stop---`without parameters---stops the Monitor.

system

□CMD="cmd"

Pass command `cmd` to operating system for execution.

8 rvs[®] Parameters

The function of the rvs[®] monitor can be influenced by the rvs[®] parameters. Their possible values and how to use them are described in this chapter. How to work with rvs[®] parameters is described in chapter 7.5 "Work With rvs[®] Parameters".

The values of the rvs[®] parameters can be set in the file `$RVSPATH/init/rdmini.dat`.

You can also use the Operator Console (`rvscns`) to set the parameters (valid only for a session).

Syntax:

`setparm PARM=VALUE`

Example: `setparm ODTRACELVL=3`

8.1 rvs[®] Parameters' Overview

The execution of the Monitor and its related components may be influenced by changing parameter values.

ACTPCOUNT (only parameter for ActivePanel)	the interval after which the statistical information about the active lines will be updated; the units are percentage of the actual filesize default: 10
AECHECK	check authority to execute (Monitor internal) command default: 0 (turned off)
BBCREATE	creation of user notifications (BB command) default: 0 (turned off)
BBPRIO	priority of user notifications (BB command) default: 90
BRICKOFTPTI	Time in seconds to wait for data by BRICK ISDN Adapter default: 20
CDWAIT	time in seconds (0-5) before executing an OFTP C hange D irection after receiving a file default: 1 (1 second)

CMDDELETE	<p>remove each command and its related entries from database as soon as command ends or is being deleted</p> <p>default: 1 (turned on)</p>
CNSMSGs	<p>IDs of LOG messages to be sent to operator console. The following message codes are defined:</p> <ul style="list-style-type: none"> A action B security E error I information L linedriver O ODETTE R report S severe error W warning + long messages <p>default: BEILORSW+</p>
CODEIN	<p>input code of a local file when creating a send entry</p> <p>default: A (for ASCII)</p>
CODEOUT	<p>output code of the file to be send. Should be the local code of the remote host; on UNIX/NT systems it is A ASCII; on OS/400 and OS/390 E EBCDIC.</p> <p>default: X is LOCAL_CODE on the remote host.</p>
DTCOⁿnn	<p>wait periods until an unsuccessful connection attempt is repeated.</p> <p>nn is the number of unsuccessful attempts (CONTRETRY in SK). There need not be a parameter for all values of nn; if a particular one is not defined, the next smaller one that is found will be used.</p> <p>format: MM/DD/YY HH:MM:SS</p> <p>defaults: increasing time intervals, so that rvs® will not be kept busy trying to reach a station that may be having hardware problems. For longer wait periods, minutes have been added</p>

to the defaults, so that retries will not occur at precisely the same time:

DTCONN01 "00/00/00 00:01:00"

DTCONN02 "00/00/00 00:02:00"

DTCONN03 "00/00/00 00:03:00"

DTCONN05 "00/00/00 00:05:00"

DTCONN07 "00/00/00 00:07:00"

DTCONN10 "00/00/00 00:10:00"

DTCONN15 "00/00/00 00:15:00"

DTCONN20 "00/00/00 00:20:00"

In addition, **DTCONN01** is the wait period for all other rvs[®] commands.

EERP_IN

Wait for a receipt from the partner

NEVER partner does not send EERP, so a send request ends with the correct transmission without waiting for the receipt

NORMAL wait for receipt, end send request when receiving receipt

Default: **NORMAL**

EERP_OUT

Handling for sending a receipt

NEVER partner does not expect EERP, so don't create a receipt.

IMMEDIATE create a receipt and start a session, if no session is available.

NORMAL create a receipt, but wait for a session to transmit (suggested)

SYNC force transmission of a receipt (EERP) for a received file in the same session in which the file was received. The connection is not closed unless the EERP is created (after successful file delivery). The rvs parameter **SYNCDL** holds a time value in milliseconds that shall be waited before it is checked whether the EERP is ready to be sent. The number of wait periods is set by the rvsX parameter **SYNCTO**. The rvsX parameters **SYNCTO** and **SYNCDL** should be configured in `$RVSPATH/init/rdmini.dat`.

Example: If **SYNCDL=400** and **SYNCTO=5**, a time span not exceeding 5 times 400ms is

waited until the connection is closed. If in this time the EERP is created, it is transmitted, and the connection is closed immediately.

HOLD create a receipt, but do not send it. When a receipt is released, send it in the next session.

HOLD_IMMED create a receipt, do not send it. When a receipt is released, create a session and send it immediately.

Please read the chapter 3.1.7 for more information about releasing the receipts in the status **HOLD** or **HOLD_IMMED**.

Default: **NORMAL**

Note: Default for station table (OP parameters) is **IMMEDIATE** (see chapter 3.1.6).

FORCEDEND

Halting of the Monitor with Monitor-Stop: Immediate cancellation, even if the transmitter and receiver are active.

NOTE: If the parameter is set at "1" the Monitor will immediately stop.

default: **0**

IEPRIO

priority of IE-commands

default: **50**

INITCMDS

execute initialization commands

default: **1** (turned on)

IZPRIO

priority of IZ commands

default: **40**

JSERRHOLD

rvs analyses this parameter when a send entry fails and rvs launches a job start after send attempt. The value of this parameter decides whether the send entry should be hold or not.

If the parameter has the value **Yes** rvs holds the send entry (sets the status to „hold“).

If the parameter has the value **No** rvs tries to finish the send entry.

Standard: **No**

KEEPDAYS

number of days, after which deleted and ended commands and their related information may be discarded during database cleanup

	default: 7
LDSNPRIO	send priority for big files. Possible values: 1 - 100. If the value is smaller, the priority is higher. This parameter should be used with parameters SDSNPRIO and SDSNMAX . If LDSNPRIO is higher than SDSNPRIO (default), the smaller files have priority. default: 50
LID	local station ID default: supplied during database initialization
LITRACELVL	request line tracing (between OFTP and network): 0 no tracing 1 minimum tracing (line driver events etc.) for station specified in parameter SIDTRACE 2 detailed tracing (incl. hex dump of data) for station specified in parameter SIDTRACE 3 detailed tracing for all stations. default: 0
LMPRIO	priority for LOG messages as external LM commands default: 20
MAXCMD	max external commands read once default: 10
MAXRECL	maximum record length for data sets with record format F or V to be received default: 99999
MAXSENDERS	maximum number of concurrent Senders. If MAXSENDERS=0 , no sender will start; default: 1
MAXX25RCV	maximum number of concurrently active or prestarted "listening" receiver processes for X.25 native communication default: 0
MSGPRIO	send priority for operator to operator messages default: 60

NUMRLOGS	number of generations of <code>rlog.log</code> files default: NOLIMIT
NUMRLSTATS	number of generations of <code>rlstat.log</code> files default: 0 , no limit
OCREVAL	ODETTE credit value = window size of OFTP: Maximum number of sent blocks without confirmation default: 99
ODTRACELVL	request line tracing (between sender and OFTP): <ol style="list-style-type: none"> 0 no tracing 1 minimum tracing (request names, only) for station specified in SIDTRACE. 2 detailed tracing (parameter values etc.) for station specified in SIDTRACE. 3 detailed tracing but for all stations. default: 0
OEXBUF	maximum ODETTE exchange buffer size in bytes (the largest ODETTE cmd (SFID)); this parameter can be configured by customizing ODETTE Parameter in the OP table, too (see chapter 3.1.6). default: 2000
OKPRIO	priority for operator commands default: 10
ORETRY	indicates the ODETTE error group for which a retry will be initiated after a request has been interrupted. It is a bit field from left to right: <ol style="list-style-type: none"> 1 – retry will be initiated; 0 – retry won't be initiated. default: 10111000111011111111 The bits stand for: <ul style="list-style-type: none"> • 1 – transmission is interrupted • 2 – file not found or can't be opened • 3 – file can not be read • 4 – „File size is too big“ in SFNA with retry is allowed

- 5 – „ Unspecified reason“ in SFNA with retry is allowed
- 6 – „ File size is too big“ in SFNA with retry is not allowed
- 7 – „ Unspecified reason“ in SFNA with retry is not allowed
- 8 – „ File size is too big“ in EFNA
- 9 – „ Invalid record count“ in EFNA
- 10 – „ Invalid byte count“ in EFNA
- 11 – „ Access method failure“ in EFNA
- 12 – „ Unspecified reason“ in EFNA

OTIMEOUT	ODETTE time-out value (in seconds) default: 600
QEPRIO	priority of QE commands default: 30
QSPRIO	priority of QS commands should lie between MSGPRIO and SDSNPRIO default: 30
RECVBLOCKS	number of buffers or records that the Receiver writes before closing temporary data set. default: 1000
RLCOMAXSIZE	maximum file size for consol messages rlco.log default: 1000000
RLDBMAXSIZE	maximum filesize for logging of database actions rldb.log default: 1000000
RLOGMAXSIZE	maximum filesize for log messages rlog.log default: 1000000
RSTATMAXSIZE	maximum filesize for statistical logs rlstat.log default: 0 , no limit
ROUTING	Sometimes it is advantageous not to allow OFTP routing. This is possible by setting the rvs parameter ROUTING for single stations in the OP table. Using the same parameter in the file \$RVSPATH/init/rdmini.dat, you can

suppress/allow it or for all partner stations.

I (means IN): the incoming file transmission from the partner (e.g. XXX) to the remote partner (e.g. REM1) via our local station (e.g. LOC) is permitted (XXX → LOC → REM1); not permitted is the outgoing routing e.g. for the partner REM2 via REM1 (LOC → REM1 → REM2).

O (means OUT): partner stations can't use your local station as router. Permitted is the outgoing routing e.g. for the partner REM2 via REM1 (LOC → REM1 → REM2). Not permitted is: the incoming file transmission from the partner e.g. XXX to the remote partner e.g. REM1 via our local station e.g. LOC (XXX → LOC → REM1).

B (means BOTH; IN and OUT): normal OFTP routing

N (means NEVER): routing in both direction IN and OUT forbidden.

default: **B**

SCPRIO

The frequency of the checking the ServiceProvider output by the rvsX monitor (in the directory SPOUTDIR (e.g. /home/rvs/temp/out), see the file \$RVSPATH/rvsenv.dat for the value of SPOUTDIR.

default: **10**

SDSNMAX

maximum size for a file to be considered short (in units of 1024 bytes)

default: **100**

SDSNPRIO

send priority for small files.

default: **40**

SECURITY

This parameter specifies the usage of the data encryption. It can be set for the corresponding station in the OP table or as rvsX global parameter in RVSPATH/init/rdmini.dat (then it is valid for stations without own SECURITY entry).

NO Encryption isn't possible. If a send job

requests encryption the job is cancelled accompanied by an error message.

OPT Encryption is optional and may be set by each send job.(see the chapter 5.4.1 about Add.parameter)

FORCED Encryption is enforced. If a send job does not switch on encryption, a warning message is created and encryption is switched on. If the partner station sends an unencrypted file the reception is denied. A send job for a partner station is handled corresponding to the **SECURITY** entry for this station. It is not important whether a partner station is a neighbour station or is reached via routing.

More about encryption please read in chapter 9.

default: **OPT**

SEENDBLOCKS number of buffers or records that the Sender sends before looking at **FORCEDEND** again.

default: **1000**

SEPRIO priority for new **SEs** should be at least as high as the highest priority of what can be transmitted by a **SE**

default: **50**

SIDTRACE ID of station that shall be traced (if **LITRACELVL** or **ODTRACELVL** are set min. to **1** or **2**).

default is " " (3 blanks).

If you need to trace incoming data, **SIDTRACE** must be set equal to the local station ID, **LID**.

SLEEP Monitor suspension time in seconds when there is no work to do, the Monitor waits this period of time, before checking, whether a command waits processing.

default: **30**

SNARCV enable start of SNA Transaction Program automatically on incoming calls:

0 no Transaction Program will start

1 a Transaction Program starts

default: **0**

SSCREATE creation of a send statistics record for each

	transfer attempt
	default: 0 (turned off)
STATISTICS	creation of a send statistics record in the statistics log file (<code>rlstat.log</code> for rvsNT and rvsX)
	<ul style="list-style-type: none"> 0 no statistics log file 1 short form 2 detailed form of statistics 3 short form of statistics inclusive routed transfers 4 detailed form of statistics inclusive routed transfers 5 new parameters such as fileformat, state of transmission, numer of dial tries 6 statistics about deleted entries (by the user), too 7 includes all of '6' inclusive routed transfers
	default: 2 (detailed statistics turned on)
SYNCDL	Please read about parameter EERP_OUT in this chapter.
	default: 500
SYNCTO	Please read about the parameter EERP_OUT in this chapter.
	default: 120
TCPIRCV	maximum number of (concurrently) prestarted "listening" processes for TCP/IP communication:
	<ul style="list-style-type: none"> 0 no TCP/IP receiver will be started n TCP/IP receiver will be started
	default: 1
TIMESTAMP	creation of a timestamp to destinguish data sets with the same dsname
	<ul style="list-style-type: none"> 1 000-999 (counter for MS DOS file names) 2 000000-999999 (counter) 3 Thhmmss (Time) 4 Dyymmdd.Thhmmss (Date and Time)

- 5 Thhmmssmsms (Date and Time in milliseconds)
default: **2** (only time)
- TMAXCON** TCP/IP maximum number of connections
0 (no limit)
default: **0** (no limit)
- TSTODPRCT** percentage of non-error returns from ODETTE simulation program when rvs[®] runs in testmode; -1 requests prompting for return values.
default: **90**
- VDSNCHAR** range of allowable characters to be transferred within an ODETTE transmission:
- **ALL**: no restrictions
 - **OFTPUNIXS**: all capital letters, digits and the special characters . -
 - **UNIX**: all letters, digits and the special characters # _ - + .
 - **ODETTE**: all capital letters, digits and the special characters () - . / &
 - **CHECK_RE**: same as ALL, but it is necessary that a RE exists
- default: **ODETTE**
- VFTYP** the way how data sets (with a fixed or variable format) will be converted.
- V** rvs[®] internal format, only useful for rvs[®] for variable and fixed formats
- S** format of ft-SINIX, useful also for ft-SINIX
- T** text format, each line is terminated by NI (line feed); each line is converted into one output record. The record length is defined in **MAXRECL**. Please see Reference manual, chapter "How to work with rvs Batch Interface", section "Command SEND" for more information how to use the parameters **MAXRECL** and **VFTYP**.
- default: **V** (rvs[®] internal format)

XMCREATE creation of LOG messages with detailed information about what was transferred from and to whom after each successful data set send or receive

default: 1 (turned on)

8.2 rvs® Parameter Values

rvs® contains a number of optional and security related features which you may not need (all the time) at your installation. When activated, these features consume computer resources (processor time and disk accesses) and thus may dramatically influence performance of rvs® components.

As an example, consider transmission of a large data set. To be able to resume transmission after a line failure without having to start at the beginning of the file all over again, the Receiver periodically closes the incoming data set, and both Sender and Receiver store the number of transmitted bytes or records in the database. The frequency of these actions is determined by parameters **SENDBLOCKS** and **RECVBLOCKS**.

Reopening and positioning a large data set involves quite a number of disk access operations and therefore is very time consuming⁵. So, if most of your communication lines are very stable, you will want to set these parameters to very large values effectively disabling the restart feature of rvs®.

If, on the other hands, most of your lines tend to break down every few minutes, you will want to make sure that whatever has been transmitted once, will not have to be transmitted again. Note, that a large value of **SENDBLOCKS** may also increase the time before Senders terminate after Monitor has been stopped.

Defaults have been chosen, so that rvs® will work securely and with most options enabled.

8.2.1 Safety, Resource Consumption and Performance

Besides **RECVBLOCKS** and **SENDBLOCKS** which have been discussed above, there are several other parameters that influence the balance between safety, resource consumption, and performance.

OCREVAL (recommended window size 99) and **OEXBUF** (recommended size 2048 bytes) influence the overhead incurred by the ODETTE protocol; the larger these values, the less overhead, but the more memory will be required for Sender

⁵ Transmission time for a 4.5 MB data set between two OS/2 nodes was reduced by about a factor of 10 (from more than an hour to a few minutes) by changing the values of these parameters from `10' to `10000'.

and Receiver. These values may be negotiated down at the start of each transmission, so that unilateral changes may have no effect. What you really determine is the maximum amount of memory you are willing to allocate to ODETTE.

Searching in a large database generally takes longer than looking for something in a small one; a larger database, however, retains more information on completed transmissions. **KEEPDAYS** determines the number of days you want to keep information about ended or deleted transmissions (unless you use cleanup days=n, explicitly specifying the retention period in the command itself).

For **CMDDELETE=1**, all related entries will be removed physically from the database when a command ends or when it is (logically) deleted. This keeps the size of the database as small as possible. If you choose this option, you should leave **XMCREATE** at its default value ('1'), so that detailed LOG messages will be created after sending or receiving a data set and all users should have access to the log data set (\$RVSPATH/db/rlog.log) to be able to look at these messages, because the dialog interface will be unable to display any information about completed transfers. Consider using this option for continuous unattended operations.

The Monitor's reaction time to new events is determined by **SLEEP**; this may influence for example, how long it takes, before the Monitor starts acting on an operator command. **SLEEP** is the period of time (in seconds) that the Monitor is suspended when there is nothing to do for it and the longer you choose this period, the less it will interfere with your other applications, but the longer you may have to wait, before it starts processing your requests. The shorter you choose this period the higher is the unproductive overhead produced by scanning the database when there is nothing to do.

The time until the Monitor restarts an unsuccessful or aborted transmission is determined by the **DTCNNxx** parameters. The smaller these values, the sooner the transmission will start after the line is up again but the more computer time may have been wasted on unsuccessful attempts until the line is restored.

8.2.2 Limit Number of Concurrent Senders

If your system is very busy or when you know, that one or more of your neighbors cannot accept more than a few incoming calls at the same time, then you want to limit the number of Senders that rvs[®] is allowed to execute at the same time.

MAXSENDERS tells MasterTransmitter *rvsxmt* how many Senders may run concurrently. When this number has been reached, it waits until a Sender terminates before starting the next one. If **MAXSENDERS** is set to **0**, no Sender will be started at all. This is useful if only the partner station should establish the connection and get the queued data sets. Use the operator command *activate*, to send data to a specific station even if **MAXSENDERS** is set to **0**.

8.2.3 Limit Number of Concurrent X.25 or ISDN Receivers

You must specify the number of concurrently active X.25 and ISDN receivers. A small number is adequate for low traffic, a higher number is required if you must be able to receive data on several connections in parallel. However, there cannot be more X.25 receivers, than virtual channels are available on your X.25 multichannel or, in case of ISDN, there cannot be more receivers than B-channels are available. Because Senders also occupy virtual channels or B channels in ISDN respectively, the number of concurrent receivers should be limited to half the total number of channels.

MAXX25RCV tells MasterTransmitter rvsxmt how many Receivers must run concurrently. It prestarts as many X.25 (ISDN) receivers as indicated. If a receiver terminates, MasterTransmitter will start a new receiver, which in turn will wait for incoming calls. **MAXX25RCV**, if set to zero, prevents any incoming X.25 or ISDN traffic. If only SNA-LU6.2 or TCP/IP is used, it must be set to zero.

On product systems, if **MAXX25RCV** is greater than 1, you have to define additional entries in the X.25 routing table (see chapter 3.1.9 "X.25 native Communications XP").

8.2.4 TCP/IP Receiver

If you want to communicate via TCP/IP, rvs® has to start a TCP/IP receiver, which waits on incoming calls. You must set the value of the parameter **TCPIPRCV**. If only LU6.2 or X.25/ISDN is used, it must be set to zero. If a TCP/IP receiver accepts an incoming call, MasterTransmitter will start a new receiver on the same port, which in turn will wait for incoming calls. So, on each port, you can accept as many calls as indicated by the values MAX_IN for your local station in your stationtable.

8.2.5 SNA Receiver

To make the rvs communication program start as Transaction Program, you have to set the value of the parameter **SNARCV** to 1. If only TCP/IP or X.25 is used, it should be set to zero. For further informations about SNA Transaction Programs see the 3.1.7.

8.2.6 Optional Features

Providing these optional services takes time and uses up disk space; so, you may wish to turn them off, if you do not need them.

AECHECK is a flag which tells the Monitor, to check whether the originator of the command currently being processed has the authority to issue this particular command. In a (future) multi-console environment, this could be used to prevent

certain consoles from stopping the Monitor, for example. Currently, this feature is not fully supported, so **AECHECK** should remain at **0** (turned off).

Statistics records will be created for every attempted transfer when flag **SSCREATE** is turned on (**SSCREATE=1**). These records contain the station ID of the neighboring rvs[®] node as well as time and completion code of the attempted (or completed) transfer. **SSCREATE=0** prevents generation of these records. Currently, no utility to analyze these records is provided.

XMCREATE (create xfer message) controls generation of detailed information about successful transfers in the system log (`$RVSPATH/db/rlog.log`, see chapter "Representation means" for the detailed description of **\$RVSPATH**). If **XMCREATE=1** (the default), a log message will be written, whenever a data set is successfully sent to a neighboring node (even before a receipt has been received), whenever a send entry completes (after receiving receipts from all recipients), and whenever a data set has been delivered to a local user. **XMCREATE=0** suppresses generation of these LOG messages.

When communication errors occur, helpful trace information can be found in the trace data sets, if **LITRACELVL** and **ODTRACELVL** are larger than zero. Tracing can dramatically reduce performance because a lot of data has to be analyzed, formatted and written into the trace file. For normal operations, tracing should be turned off, i.e. both parameters should be set to **0**.

CNSMSGs controls, which LOG messages are echoed to the operator console. All messages, whose code letter is included in the character string value of **CNSMSG**s are written to the console (all messages are always logged, independent of the value of **CNSMSG**s). The additional message types 'O' (ODETTE), 'L' (Linedriver) and + (for long messages) can now also be used.

STATISTICS controls the creation of the statistic log file. **STATISTICS=1** creates the file (`$RVSPATH/db/r1stat.log`). It contains a line for each sent or received file with name, date, time and sender/receiver sid. **STATISTICS=2** creates the same file, but with extended information (e.g. the file name for the transmission (virtual file name), the file size and command numbers for SE, SK or IE, IZ). **STATISTICS=3** is the same as **STATISTICS=1**, but routed filetransfer will be logged, too (i.e. **SID** of destination station and **SID** of source station). **STATISTICS=4** is the same as **STATISTICS=2**, but routed filetransfer will be logged too. **STATISTICS=5** means a detailed output in (`$RVSPATH/db/r1stat.log`). with new parameters such as file format, state of transmission and number of dial tries. **STATISTICS=6** produces a more detailed output about deleted entries (by the user) with the cause of deletion (if specified with `delcmd`). **STATISTICS=7** includes all of **6** with information about routing. **STATISTICS=0** prevents the creation of this file.

9 Configuration of Encryption: Key Administration

This chapter contains the description of the key administration. It describes how to generate, import and distribute keys as well as how to list and delete imported keys.

In order to use encryption please take these steps:

1. Check the rvs[®] license key. If the module **Encryption** was purchased the rvs[®] license key (`$RVSPATH\init\rdkey.dat`) contains the module **Y**.
2. Create an encryption key pair for your local station (use the tool `genKey`, see chapter 9.1.1 "Creation of own Private and Public Key").
3. Import the own keys (private and public) into rvs[®] (use the tool `rvskeyimp`, see chapter 9.1.2 "Importing Keys (`rvskeyimp`)"). Please handle the private key file very carefully to ensure your privacy.
4. Send the own public key file to the partner stations that shall participate in encrypted file transfer. You may use rvs[®] itself to perform this (use the tool `rvskeydst`, see chapter 9.1.3 "Distribution of Keys (`rvskeydst`)").
5. Get a public key file from each of your partner stations that may participate in encrypted file transfer. Import the keys into rvs[®] (use the tool `rvskeyimp`, see chapter 9.1.2 "Importing Keys (`rvskeyimp`)").
6. Check rvs[®] key configuration to ensure availability of the local private key and of the partner public keys using the rvs[®] tool `rvskeylst` (see chapter 9.1.6 "Listing of Imported Keys (`rvskeylst`)").

Encryption may be switched on for each send job separately (see the chapter 5.4.1 `Add.parameter`). You can also handle encryption with the parameter `SECURITY` (global in the file `RVSPATH/init/rdmini.dat` or for each station in the `OP` table. Please see the chapter 3.1.6 for more information about the parameter `SECURITY`.

Encryption is handled automatically when receiving files. The precondition is that you have done all 6 steps mentioned above.

Encryption functions directly between a sender and a receiver. The encrypted files are routed without problems via routing stations.

9.1.1 Creation of own Private and Public Key

The encryption function of rvs needs private/public keys generated by the utility program `genKey`. This is an easy to use program, which creates randomly an new key pair at any call.

Command line:

The following program parameters are possible:

```
genKey
```

```
[-chefmopstv] [--768] [--1024] [--2048]
[--owner] [--creator] [--from] [--to]
[--help] [--size] private_key public_key
```

The parameters `private_key` and `public_key` stand for the files where the generated keys will be saved. Only this two parameters are required to produce a new keypair. The other parameters are optional.

Example:

```
genKey -h
genKey --h
```

These two calls enables to you to show the help program.

Optional parameters:

Parameter	Description	Default
-h --help	print this message	
-c --creator <i>string</i>	creator of the files Examples: -c gha -- creator gha	'unknown'
-o --owner <i>string</i>	owner of the files	'unknown'
-f --from <i>string</i>	first day of valid period (TT.MM.JJJJ YYYY-MM-DD) You have a choise between the german and the international version.	today
-t --to <i>string</i>	last day of valid period (TT.MM.JJJJ YYYY-MM-DD)	
-p --per <i>string</i>	valid period d D t T=day , m M=month ,	3 month

	j J y Y=year, h H=hours, i I=minutes, s S=seconds You must choose one time unit. Examples: 1m (1 month) or 5D (5 days)	
-s --size <i>value</i>	keysize in bit (the keysize must be at most 2048 bit and divisible by 8)	1024
--768	creates a keypair with 768 bits	
--1024	creates a keypair with 1024 bits	
--2048	creates a keypair with 2048 bits	
-e	use the exact current time (normally the valid period starts at 00:00:00 UTC)	
-m	print process information (not reasonable on batch systems)	
-q	be quiet	
-v	be verbose	

Examples:

```
genKey TC2private TC2public
(Generates a private key named TC2private and a public key named
TC2public. The length of the keys will be 1024 bit and they will be valid three
months beginning with the actual day.)
genKey -c Fischer -f 01.06.2001 -t 01.12.2001 -s 2048
WO1.pri WO1.pub
(Generates a key pair with the creator mentioned as Fischer and with a size of
2048 bit. The valid period starts on June, 1 2001 and ends on December, 1
2001.)
```

The generated keys must be imported into the rvs[®] data base.

9.1.2 Importing Keys (rvskeyimp)

The tool rvskeyimp imports a key into the rvs[®] data base.

To import a key it is necessary to provide the name of the file containing the key with full path and the ID of the station (SID) for which the key shall be used. The tool rvskeyimp reads the file containing the key, copies it into the directory

containing all imported keys (see parameter `KEYDIR` in `rvsenv.dat`) and writes all necessary informations into the `rvs`[®] database (Table SV: S=Keys, V=Administration).

Command line:

Usage:

```
rvskeyimp [-?] -i <FILE> -s <SID>
```

Options:

- i <FILE> import key from file <FILE>, whereat <FILE> must contain the full path
- s <SID> the station ID for which the key will be imported

Example:

```
rvskeyimp -i /home/rvs/usrdat/A15A13.pri -s LOC
```

```
rvskeyimp -i /home/rvs/usrdat/A15A13.pub -s LOC
```

```
rvskeyimp -i /home/rvs/usrdat/A15A13.pub -s RSL
```

9.1.3 Support for public key certification

With this feature a certification request for the own public key can be generated following PKCS #10. This may be required e.g. if the certificate shall be deposited in a trust center. The certification request is created using the tool `rvskeyreq`:

Usage:

```
rvskeyreq [-?cod] outfile
```

Options:

- ? Shows usage
- c <CN> Common Name (mandatory); Common Name is the keyword for the search of the certificate in a PKI.
- o <ORG> Organisation (mandatory); this is the name of your company, this parameter would not be used for the search in the PKI, so you can write here the short name of your company.

- `-d <Dumpfile>` Auxiliary dump file, includes informations about the PKCS#10 request such as Common Name, Organisation and the own public key.
- `outfile` Output file; contains the same informations as the dump file, but base64 encoded.

Example:

```
rvskeyreq -c "VW_OFTP 000134350GEDASRVS33" -o gedas -d
/home/rvs/usrdat/dump /home/rvs/usrdat/request
```

This example creates a dump file `dump` and a certification request `request`. In the file `dump` are the parameters `CN` (Common Name), `ORG` (Organisation) and your public key. The same parameters are in the file `request`, but base64- encoded.

The file `dump` could look as follows:

```
CN (Common Name) = VW_OFTP 000134350GEDASRVS33
ORG (Organisation) = gedas
```

Public Key:

```
00:bd:79:69:5d:96:a7:16:c6:02:e1:69:d2:14:53:af:
98:47:9e:26:56:e7:f4:18:fd:8c:77:71:cc:ef:c5:6e:
65:81:9a:2f:9b:2d:ed:c7:b9:b5:4d:24:11:09:e6:53:
7e:ba:4a:8a:eb:db:84:18:ab:c2:78:2e:fe:de:17:c7:
7d:65:f5:98:e5:89:20:ae:83:cb:7d:68:05:e6:69:90:

10:50:da:f5:a3:40:f3:af:bd:60:ee:26:05:c5:f9:99:
99:8a:c0:9d:f6:de:64:0d:cb:e5:a4:54:69:8f:91:2b:
ed:1a:64:42:e7:42:a4:34:92:5d:fd:cb:94:8f:00:4c:
8f:
```

The file `request` could look as follows:

```
-----BEGIN CERTIFICATE REQUEST-----
MIIBeTCB4wIBADA8MQ8wDQYDVQQKFgZSV1NDQzExKTAnBgNVBAMWIFZXX09G
VFAGtZAwMTMwMDMyMTBHRURBUy0tLVJWU0NDMIGfMA0GCSqGSIb3DQEBAQUA
A4GNADCBiQKBgQC9eWld1qcWxgLhadIUU6+YR54mVuf0GP2Md3HM78VuZYGa
L5st7ce5tU0kEQnmU366Sorr24QYq8J4Lv7eF8d9ZfWY5YkgroPLfWgF5mmQ
EFDa9aNA86+9Y04mBcX5mZmKwJ323mQNY+WkVGmPKSvtGmRC50KkNJjd/cuU
jwBMjwIDAQABMA0GCSqGSIb3DQEBBQUAA4GBAEP+42YhF7fyRNZuOPHCQ3sx
/oTzjjN+pPqaqfCrVdyciKiI+zwbErsb53JaLMQYXTLixdHxcnoH2xxAVYG5
f0MB23TnZrCJAp8Xw3Kn4i6vF4+YTUYT8ZdHYyBEGOKcltVtYOHOQYcUVA8h
iL60onHlbsKxuQNZjLZxeKiNouIJ
-----END CERTIFICATE REQUEST-----
```

9.1.4 Distribution of Keys (`rvskeydst`)

To enable the communication partner to receive and decrypt data, which have been encrypted by the sender, it is necessary to provide the public key to the partner. This can be done by e-mail, mail, fax, disk or as well with `rvs`[®].

The tool `rvskeydst` sends a key file to a communication partner. It needs as parameters the file name of the file containing the key and the ID's of the stations (SID) the key will be send to. The station ID's can be either handed over as parameter or in the form of an input file (one line for each station ID).

Usage:

```
rvskeydst [-?] -f <FILE> -s <SID> | -l <FILE>
```

Options:

- f <FILE> the file containing the key, <FILE>
must contain the full path
- s <SID> the ID of the station to which the key will be send
(repeatable)
- l <FILE> the file containing a list of station ID's

Example:

```
rvskeydst -f /home/rvs/keycreate/C45PUB.pub -s R11 -s A34 -s  
gedas
```

```
rvskeydst -f /home/rvs/keycreate/C45PUB.pub -l  
/home/rvs/list/sendlist1.lst
```

9.1.5 Deletion of Imported Keys (`rvskeydel`)

The tool `rvskeydel` deletes keys that previously were imported into the `rvs`[®] database. The key gets deleted in the database and the file in the key directory will be removed.

Command line:

Usage:

```
rvskeydel [-?ldur]
```


Options:

- l generates a list of all key entries

- d KEYID deletes key with the ID KEYID

- u removes the public key

- r removes the private key

The option `-l` generates a list containing delete statements for all keys in the database. The output contains two lines for each key, one with all key data and the other one with the corresponding deletion statement.

To use this list to delete keys the output has to be redirected into a file. All deletion statements are commented out. Remove the comment mark (`REM`) of the keys that are supposed to get deleted and invoke it from the command line.

Example of a generated key list:

```
# SID=LOC DATEBEGIN=2001/01/25 00:00:00 DATEEND=2001/04/25 02:00:00 ...
# rvskeydel -d 1074002581 -r
# SID=LOC DATEBEGIN=2001/01/25 00:00:00 DATEEND=2001/04/25 02:00:00 ...
# rvskeydel -d 1074002581 -u
```

Example for usage:

```
rvskeydel -l
    (Lists all key entries.)
```

```
rvskeydel -d 103456734 -u
    (Removes the public key with the ID 103456734.)
```

9.1.6 Listing of Imported Keys (`rvskeylst`)

The tool `rvskeylst` lists all keys that have been imported into the database with their complete data. This command doesn't support any options.

Usage:

```
rvskeylst
```

Example of result:

```
SID=WO1 BEGIN=2001/02/22 00:00:00 END=2001/05/22 01:00:00 FILE= ...
```

SID=LOC BEGIN=2001/02/22 00:00:00 END=2001/05/22 02:00:00 FILE= ...

9.2 Configuration of Offline Compression

If the module **Offline compression** was purchased it is enabled by the rvs[®] license key. Offline compression is enabled when the key file `rdkey.dat` (which is located in the rvs[®] `init` directory) contains the module **Z**.

There are none additional configuration steps required for offline compression. Offline compression may be switched on for each send job separately. It is handled automatically when receiving files.

10 Code Conversion

You are probably aware that text files are stored on most systems in one of two computer codes, namely ASCII (American National Standard Code for Information Interchange) or EBCDIC (Extended Binary Coded Decimal Interchange Code). ASCII is the standard code for UNIX and DOS/Windows Systems. While the assignment of digits and letters of the Latin alphabet is standardized within each of these two code families, special characters (like square brackets '[']') or national language characters (like accented letters or umlauts) may be assigned to different codes in different code pages within a family.

Odette Standard, OFTP and so rvs[®] distinguish between **text** files (format T) and **non-text** files (formats F, V and U).

Text files are always transmitted in ASCII (**ODETTE** Protocol) and automatically delivered in the local code of the target system.

Non-text files are transmitted as is without conversion, unless you specifically request code conversion by specifying INPUT CODE and OUTPUT CODE when creating the send entry.

Example1 (Text file):

Direction: rvsX ⇒ rvsMVS

On the UNIX systems text files are represented in ASCII and on the OS/390 System in EBCDIC Code.

A rvsX station will send a text file to a rvsMVS station. A text file must be sent in the format Text. (rvsdia → send a dataset → format=T(text). This file will be transmitted as an ASCII file (**ODETTE**) and will be converted at the receiver (rvsMVS Station) into EBCDIC.

Direction: rvsMVS ⇒ rvsX

When sending a text file from an OS/390 system you must define the format of the file (Format=Text), so this EBCDIC text file will be converted before sending to ASCII code, because of the **ODETTE** rules. Text files are transmitted only in ASCII format. The rvsX system will receive the text file in ASCII.

Example 2 (Non text files):

Direction: rvsX ⇒ rvsMVS

Normally an OS/390 system requires files in Format F (fixed) or V (variable) to be able to store and process them without problems. Unix systems are able to handle only files in format T (text) or U (unstructured). So, if you want to send files with rvsX to an OS/390 system, you have to convert them to the pseudo fixed or variable format before sending. This feature offers you a rvs tool `rvsut2fv`. More details about `rvsut2fv`, you can find in the Reference manual, Part III. The rvsX station would send file converted by the `rvsut2fv` in format F or V. The values of the Input and Output Code parameters should be set, too. (`rvs dia` → send a dataset → Format=F(fixed)/V(variable) → Input Code=A; Output Code=E). The Input Code is A (ASCII); Output Code E (EBCDIC).

rvs offers you the possibility of code conversion by sending and by receiving. For the code conversion by receiving, you should define a resident receive entry (see chapter 5.5.1 "Resident Receive Entries").

For the code conversion, you can use two sorts of code conversion tables:

- those, which are installed in rvs[®]
- your own conversion tables.

10.1 Automatic Code Conversion with rvs System Code Tables

The rvs[®] code conversion tables are built according to the following norms:

**ASCII: ISO 8859 Latin 1 and
EBCDIC: CECP 037**

The rvs[®] code conversion tables are `$RVSPATH/init/rtcae.dat` (for ASCII to EBCDIC) and `$RVSPATH/init/rtcea.dat` (for EBCDIC to ASCII).

The meaning of the letters in those file names is:

r rvs[®]
t translate (conversion)
c code
e EBCDIC
a ASCII

10.2 Code Conversion with User Code Tables

The rvs[®] user can use his own ASCII \leftrightarrow EBCDIC code tables instead of using the system ASCII \leftrightarrow EBCDIC code tables for code conversion.

The most important parameters for the code conversion are Input Code, Output Code and Codetable in the dialog interface rvsdia (rvsdia \rightarrow send a dataset \rightarrow add.parameter=Y \rightarrow Input Code=A/E; Output Code=A/E, Codetable=="<path and name of user code table>"). The same parameters in rvsbat are: CODEIN, CODEOUT and CODETABLE. The parameter Input Code defines the code of the file before conversion and the parameter Output Code is the code after the conversion. If you want to use your own table for the conversion, you should indicate it with the help of the parameter Codetable.

To use the user specific code tables you have two possibilities:

- You define the parameter: Input Code, Output Code and Codetable. See chapter 10.3 "How to Carry out a Code Conversion" for more details.
- You define only Input Code, Output Code and replace the system code conversion tables \$RVSPATH/init/rtcae.dat and \$RVSPATH/init/rtcea.dat with your own code conversion tables. Your own conversion tables must be named rtcae.dat or rtcea.dat, too.

10.2.1 Structure of the Code Conversion Tables

If you intend to create your own code conversion tables, these should have the same structure as rvs[®] system code conversion tables (files \$RVSPATH/init/rtcae.dat or rtcea.dat).

The rvs[®] code conversion tables have 256 decimal numbers (8 bit). The position in the table (0th position in the table is the number 0) defines the input code and the number, that is located on this position is the output code for the same number.

The following example should explain this complicated content:

To understand it, you need an ASCII code table, an EBCDIC code table and rvs[®] code tables (rtcae.dat and rtcea.dat). If you want to realize the code conversion ASCII \leftrightarrow EBCDIC for the small letter r, you would find in the ASCII code table number 114 that represents this letter. The number 114 defines the position in the code conversion table rtcae.dat of the decimal value for the small letter r in the EBCDIC table.

So, you can find at the 114th position (number 0 is 0th position) in the file `rtcae.dat` the number 153, that is the value for the small letter `r` in the EBCDIC table.

The same procedure is also valid for other letters. One more example: **A**. On the 65. position in the file `rtcae.dat` is the number 193. The number 193 is the EBCDIC decimal value for the capital letter **A**. In the opposite direction you can find on the 193th position in the file `rtcea.dat` ASCII code 65 for **A**.

10.3 How to Carry out a Code Conversion

This chapter describes how to send and receive files via menu interface `rvsdia` and via `rvsbat` with code conversion.

10.3.1 Code Conversion when Sending Files

Sending files with code conversion via `rvsdia`

◆ Automatic ASCII to EBCDIC code conversion

1. start `rvsdia`
2. choose `send a dataset`
3. enter your send parameter; at `add.parm` enter **Y** for yes
4. at `Input Code` enter **A** for ASCII
5. at `Output Code` enter **E** for EBCDIC

Result: An ASCII file will be sent as an EBCDIC file.

Note: If you partner station expects the files in format Fixed or Variable, you should convert them with the utility `rvsut2fv` into a pseudo fixed or variable format. More details about `rvsut2fv`, you can find in the Reference manual, Part III.

◆ Code conversion with user specific code conversion tables

1. start `rvsdia`
2. choose `send a dataset`

3. enter your send parameter; at `add.parm` enter **Y** for yes
4. at `Input Code` enter **A** for ASCII
5. at `Output Code` enter **E** for EBCDIC
6. at `Codetable` enter the path of your code table (e.g. `$RVSPATH/arcdire/rtcusr.dat`)

Result: A file will be sent and converted as specified in the user code table.

Sending files with code conversion via `rvsbat`

◆ Automatic ASCII to EBCDIC code conversion

1. create a job file with ASCII to EBCDIC code conversion

Example:

create a job file (e.g. `job`) that contains the following lines

```
send /c dsn="<file to send>" format=U codein=a  
(sid="<destination station>" codeout=e)
```

2. start the job with `rvsbat /ijob` or `rvsbat -ijob`

Result: The ASCII file `<file to send>` in format U will be sent as an EBCDIC file.

◆ Code conversion with user specific code conversion tables

1. create a job file with the following parameters:

Example:

create the job file (e.g. `job`) that contains

```
send /c dsn="<file to send>" format=U codein=a  
(sid="<destination station>" codeout=e codetable="<path  
and name of user code table>")
```

2. start the job with `rvsbat /ijob` or `rvsbat -ijob`

Result: A file `<file to send>` will be sent and converted as specified in the user code table.

10.3.2 Code Conversion when Receiving Files

This chapter describes how to use the code conversion for receiving files.

Using Resident Receive Entries created via `rvsdia`

◆ **Create a resident receive entry with code conversion:**

1. start `rvsdia`
2. choose resident receive entries appears.
3. at display/change/delete a resident receive entry hit **<ENTER>**.
4. to define parameter for a new resident entry hit **<F2>** or **<&>**.
5. enter your resident receive entry parameter
6. at code table enter path and name of the rvs[®] system code conversion table (`$RVSPATH/init/rtcae.dat` or `$RVSPATH/init/rtcea.dat`)
7. to create the new resident receive entry hit **<ENTER>**

Result: An ASCII file received via the defined resident receive entry will be saved as an EBCDIC file.

◆ **Create a resident receive entry with user specific code conversion tables:**

1. – 4. see **Create a resident receive entry with code conversion**
5. enter your resident receive entry parameter
6. at code table enter path and name of your own code conversion table (`$RVSPATH/arcdire/rtceaown.dat`)

Result: A received file will be converted with your own code conversion table.

Using Resident Receive Entries created via `rvsbat`

◆ **Create a resident receive entry with ASCII to EBCDIC code conversion:**

1. create a job file that creates the resident receive entry

Example:

create the job file (e.g. job) that contains

```
resentr /c dsn="<received ASCII file>" codetrans=e  
sid="<send station>"
```

2. start the job with `rvsbat /ijob` or `rvsbat -ijob`

Result: An ASCII file received via the defined resident receive entry will be saved as an EBCDIC (codetrans=e) file.

◆ **Create a resident receive entry with EBCDIC to ASCII code conversion:**

1. create a job file that creates this resident receive entry

Example:

create the job file (e.g. job) that contains

```
resentr /c dsn="<received EBCDIC file>" codetrans=a  
sid="<send station>"
```

2. start the job with `rvsbat /ijob` or `rvsbat -ijob`

Result: An EBCDIC file received via the defined resident receive entry will be saved as an ASCII (codetrans=a) file.

◆ **Create a resident receive entry with user specific code conversion table:**

1. create a job file that creates this resident receive entry

Example:

create the job file (e.g. job) that contains

```
resentr /c dsn="<received EBCDIC file>" codetrans=t  
codetable="<user codetable, e.g.  
$RVSPATH/arcdire/rtcusrdat>" sid="<send station>"
```

2. start the job with `rvsbat /ijob` or `rvsbat -ijob`

Result: A file received via the defined resident receive entry will be processed converted with the user defined code table.

11 rvsX Oracle Binding (rvsX High Performance)

In order to efficiently master the continuously growing flow of data and to increase the performance level of rvsX, rvsX 2.05 and above gives the possibility of binding to an Oracle database. The rvs[®] internal C-ISAM database is replaced with the external high performance Oracle database.

rvsX Licence: The rvs[®] licence key file `$RVSPATH/rdkey.dat` must include the letter **O** in the line **Included Components**, to enable you to start the rvsX High Performance.

11.1 Configuration of ORACLE

Conditions: In order to install rvsX High-Performance you must have a fully functioning Oracle Database, at least version 8.1.7. Further, your configuration file (e.g. `.profile` on UNIX Systems) has to be expanded with Oracle suitable definitions of environment variables.

Here are two examples of the configuration file for LINUX and AIX systems with Oracle environment variables.

LINUX

```
ORACLE_OWNER=oracle
export ORACLE_OWNER
ORACLE_HOME=/opt/oracle/product/8.1.7
export ORACLE_HOME
ORACLE_SID=rvslnx4
export ORACLE_SID
export DBID=ORA
PATH=.:$PATH:$ORACLE_HOME/bin
export PATH
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/bwa/rvs/system:
/home/bwa/rvs/lib:/opt/oracle/product/8.1.7/lib
export LD_LIBRARY_PATH
```

AIX

```
ORACLE_SID=rvsora
export ORACLE_SID
export DBID=ORA
PATH=$PATH:$ORACLE_HOME/bin:
export PATH
```

Two configuration options are offered:

1. rvs[®] and Oracle work on the same computer
2. rvs[®] and Oracle work on different computers

For 2.: When rvs[®] and Oracle are working on different computers, the communication between both systems takes place via a LAN, that must support both systems. Here the normal Oracle Client/Server solution would be used.

Server: The Oracle Server Software must be installed and configured on the database server. Further the so called LISTNER must be configured on this server.

Client: The rvsX OFTP server and Oracle client must be installed and configured on the client.

11.2 Configuration of rvs[®]

The following steps are necessary:

11.2.1 rvs[®] and Oracle Working on the Same Computer

The following variable definitions are to be added to the rvs[®] environment file \$RVSPATH/rvsenv.dat:

```
DBNAME = '<ORACLE system ID>'
DBUSER = '<ORACLE user>'
DBPSW  = '<ORACLE user's password>'
```

ORACLE system ID, ORACLE user and ORACLE user's password have been defined when installing and configuring Oracle.

Note: ORACLE user and ORACLE user's password must be identical to that used by a rvs[®] user with administrator authority.

Example (\$RVSPATH/rvsenv.dat):

```
SYSTEM      = '/home/skk/rvs/system'
DB           = '/home/skk/rvs/db'
INIT        = '/home/skk/rvs/init'
TEMP        = '/home/skk/rvs/temp'
USRDAT      = '/home/skk/rvs/usrdat'
SAMPLES    = '/home/skk/rvs/samples'
ARCDIR     = '/home/skk/rvs/arcdir'
KEYDIR     = '/home/skk/rvs/keydir'
SPFILESDIR = '/home/skk/rvs/temp'
SPINDIR    = '/home/skk/rvs/temp/in'
SPOUTDIR   = '/home/skk/rvs/temp/out'
DBLOG       = 'N'
LANGUAGE   = 'D'
DFTAUT     = '660'
MODE        = 'n'
PORT        = '2956'
SERVER     = 'rvsaix3'
DBNAME     = 'rvsora'
DBUSER     = 'skk'
DBPSW      = 'skk'
```

11.2.2 rvs[®] and Oracle Working on Two Different Computers

If Oracle is running on an external computer, this computer must be defined in the file :

```
$ORACLEHOME/network/admin/tnsnames.ora.
```

The following variable definitions are to be added to the rvs[®] environment file \$RVSPATH/rvsenv.dat:

```
DBNAME      = '<ORACLE system ID>'
DBUSER      = '<ORACLE user@ORACLE database>'
DBPSW      = '<ORACLE user's password>'
```

These variable values have been defined when installing and configuring Oracle.

Note: The same applies as in paragraph 11.2.1. ORACLE user must also be an existing rvs[®] user with administrator authority

Example (tnsnames.ora):

```
#TNSNAMES.ORA          Network          Configuration          File:
/opt/oracle/product/8.1.7/network/admin/tnsnames.ora
# Generated by Oracle configuration tools.

RVSORA.GEDAS.DE =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = rvsaix3)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = rvsora.gedas.de)
    )
  )
```

Example (extract from \$RVSPATH/rvsenv.dat):

```
* rvsX - enviroment file (rvsenv.dat)

DBNAME      = 'rvsora'
DBUSER      = 'skk@RVSORA.GEDAS.DE'
DBPSW      = 'skk'
```

12 rvs® Data Center

The present chapter describes how to operate the rvs® Data Center.

It is highly recommended that you read chapter 2.10 in the Reference Manual prior to installing rvs® Data Center in order to learn more about the technical basis and software architecture of rvs® Data Center.

12.1 Introduction

rvs® Data Center is a server farm featuring a very high fail safety and transfer capacity.

To ensure a very high system availability, a server farm comprises an array of several computers (rvs® servers in case of the rvs® Data Center). To ensure trouble-free server farm operation another server can assume the tasks of a failed server.

12.2 System requirements

As of rvs® version 3.05.00, rvs® Data Center is available for the following platforms:

- AIX 5.2.

Oracle version 8.1.7 is used as rvs® database. To ensure access to the Oracle database, Oracle client software must be installed on each rvs® server (node).

The NFS (Network File System) protocol version 3 is required to access the shared directories of the rvs® Data Center over the network.

12.3 Installation

Installation of an rvs® node is similar to a standard rvs® installation. For this purpose please read chapter 2.2 as this chapter exclusively covers the installation step specific for rvs® Data Center.

12.3.1 Installation of the first rvs® node

Installation of an rvs® node is similar to a standard rvs® installation. Perform the first seven steps as described in chapter 2.2. In step 8, “Choose installation/update path” window, you are asked if you wish rvs® paths to be automatically created. It is mandatory that you answer **n** (no) here. This is very important as the central

directories need to be installed on another computer. By default, the installer will install the rvs[®] directories in the \$RVSPATH directory on the same computer.

Having answered this question with **n** (no) you will be separately prompted for the path of each rvs[®] directory. Please note that the following rvs[®] directories must be installed in the central directory on another computer: temp, usrdat, init, keydir, samples and the ServiceProvider directories (parameters **SPFILESDIR**, **SPINDIR** and **SPOUTDIR** in the rvsenv.dat rvs[®] environment file; defaults: ../temp/temp, ../temp/in and ../temp/out).

Example:

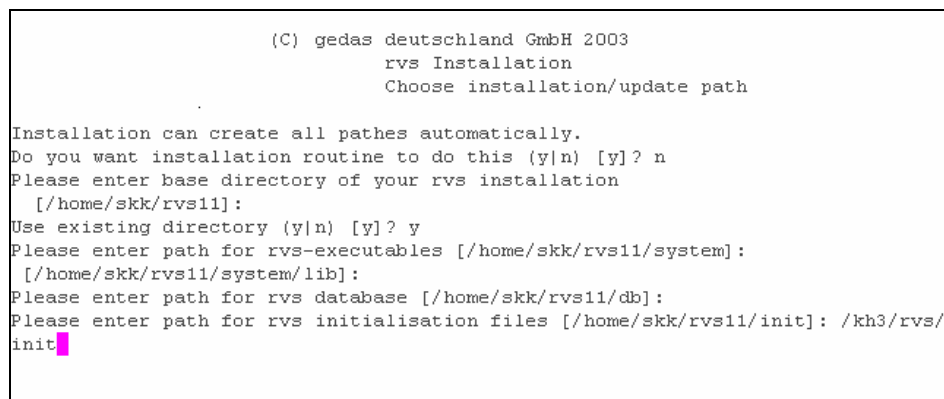
```
//computer2/central directory/init
//computer2/central directory/temp
//computer2/central directory/usrdat
//computer2/central directory/samples
//computer2/central directory/keydir
//computer2/central directory/temp/temp
//computer2/central directory/temp/in
//computer2/central directory/temp/out
```

Create the system, db, arcdir and tracedir directories on the local computer where you are running the installer.

Example:

```
//rvs nodel/rvs/system
//rvs nodel/rvs/db
//rvs nodel/rvs/arcdir
//rvs nodel/rvs/tracedir
```

The following screenshot illustrates this installation step:



```
(C) gedas deutschland GmbH 2003
rvs Installation
Choose installation/update path

Installation can create all pathes automatically.
Do you want installation routine to do this (y|n) [y]? n
Please enter base directory of your rvs installation
[/home/skk/rvs11]:
Use existing directory (y|n) [y]? y
Please enter path for rvs-executables [/home/skk/rvs11/system]:
[/home/skk/rvs11/system/lib]:
Please enter path for rvs database [/home/skk/rvs11/db]:
Please enter path for rvs initialisation files [/home/skk/rvs11/init]: /kh3/rvs/
init█
```

The present example creates the `system` and `db` directories on the local computer in the `$RVSPATH` directory (`/home/skk/rvs11` in the example). The `init` directory is located in the central `kh3` directory on a remote computer and must be accessible via NFS.

Note: All files to be sent must also be located in the central directory; the installer, however, does not offer any path variable for this.

We recommend that you accept the suggested defaults for all other queries, e.g.:

- Language [E]:
- DB log mechanism [N]
- user rights for received files [660]
- the `modus` parameter is not supported here; you must therefore accept the default [n] as well.
- rvs port number is the LogWriter port, default: 2956
- rvs logwriter server: rvs® server name
- local rvsnode [node1]: `node1` is the default for the `RVSNODENAME` parameter. `RVSNODENAME` is the name for the individual rvs® node in the rvs® Data Center.
- format of logmessages [M]: `M` is the default for the `LOGFORMAT` parameter (see chapter 12.8).
- `db name`, `db user` and `password` for ORACLE are access parameters for an Oracle database (see chapter 11).
- delay in seconds for database connection recovery [20]: This is the `DBDL` parameter (see chapter 12.7.1), the default is 20.
- maximum of tries to recover database connection [6]: This is the `DBTO` parameter (see chapter 12.7.1), the default is 6.

Note: Do not accept the default for the last query (“enter if logmessages should be stored in DB [0|1] [0]:”)! This is the `LOGINDB` parameter that must be set to 1 so that log messages can be written to the database and rvs® Client/Server can interpret them (see chapter 12.8).

Following installation, all parameters mentioned here can be found in the `rvs® $RVSPATH/rvsenv.dat` environment file.

The remaining steps required to install the first rvs® node are identical to those for a normal rvs® installation. An appropriate description can be found in chapter 2.2.

Following installation of the first node you must adapt the `farmstart/farmstop`, `nodestart/nodestop` and `rvsrestartnode` scripts according to your needs (see chapters 12.5.3 and 12.5.4).

12.3.2 Adding an rvs[®] node

Once you have completed installation of the first node you can add other rvs[®] nodes to your rvs[®] Data Center.

To begin with, install the new node like a normal rvs[®] installation (see chapter 2.3). Create all rvs[®] directories on the local computer in this step. To prevent the initial installation from being overwritten you must not assign the `temp`, `usrdat`, `init`, `keydir`, `samples`, `spfilesdir`, `spindir` and `spoutdir` to the central directory as during initial installation.

In the next step (following successful completion of normal rvs[®] installation), you must adapt the `$RVSPATH/rvsenv.dat` environment file so the present installation can also access the central directory. This means the local paths for rvs[®] directories `temp`, `usrdat`, `init`, `keydir`, `samples`, `spfilesdir`, `spindir` and `spoutdir` must be assigned to the central directory located on the remote computer, which was created during initial installation. The central directory must be accessible via NFS. In the example in chapter 12.3.1 this was `kh3`.

Following this customization you must complete the `farmstart/farmstop`, `nodestart/nodestop` and `rvsrestartnode` scripts according to your needs (see chapters 12.5.3 and 12.5.4).

12.3.3 Updating all nodes (release change)

A new release is available if the first- or second-digit rvs[®] version number was changed. An update from rvs[®] 3.05.00 to 3.06.00 or 4.00.00 is a release change.

To implement a new release you must stop all rvs[®] nodes (the entire rvs[®] Data Center). The update is to be executed on each node as described in chapter 2.3. Manual user intervention as for installation of an rvs[®] node is not required (the central directory will not be overwritten during an update).

The update will back up the database and convert it to the new version.

12.3.4 Updating all nodes (patch update)

A patch update is available if the third-digit rvs[®] version number was changed. An update from rvs 3.05.00 to 3.05.04 is a patch update. A patch update corrects errors but does not imply any interface or database structure changes.

A patch update can be executed at runtime. You must stop the node where you install the new version only. All other rvs[®] Data Center nodes can continue being active.

Perform an update of the node to be patched as described in chapter 2.3 after having stopped the node to be patched. Please note that you need not back up the old version during installation and that the database is not migrated (step 9 in chapter 2.3). A migration is not necessary and the other nodes continue using the database.

12.4 How to control fail safety

rvs® Data Center offers a number of parameters for fail safety configuration. The values for these parameters significantly affect the job processing safety.

Important note: Following installation, default values (= recommended values) are defined; these should be changed only if rvs® Data Center does not work properly.

The following new global rvs® parameters you must configure in the `$RVSPATH/init/rdmini.dat` file have been provided for this purpose:

- COMTIMEOUT
- MONTIMEOUT
- CNTMA
- CNTGC

Configure the following new parameters in the `$RVSPATH/rvsenv.dat` rvs® environment file:

- SPERRTO
- SPTIMEOUT

Parameters COMTIMEOUT, MONTIMEOUT, CNTMA and CNTGC refer to monitor activities whereas SPERRTO and SPTIMEOUT refer to Service Provider activities.

You can also set the new DBDL and DBTO parameters in the `$RVSPATH/rvsenv.dat` file; these parameters are of particular significance in case of low resources (see chapter 12.7).

Note: The time needs to be synchronized on all nodes in order to ensure correct monitor operation and log message consistency. Time synchronization is a requirement to the system where the rvs® Data Center is installed.

12.4.1 Monitor parameters

This chapter describes the parameters relevant for fail safety related to monitor activities. First the four timers are presented; they are to determine the moment when a process is to be regarded as inactive. An explanation of the CNTMA and CNTGC parameters follows. They allow the number of commands to be configured that are executed before rvs® Data Center monitoring turns active.

12.4.1.1 Timers

This chapter describes the two new and the two old timers (already present in earlier versions).

12.4.1.1.1 OTIMEOUT

This is not a new parameter; it is mentioned because its values affect the communication process (`rvscom`).

`OTIMEOUT` is the Odette timeout for activities at line level.

It must be set in the `$RVSPATH/init/rdmini.dat` file.

Recommended value: 30 (seconds) for ISDN, 600 (seconds) for X.25. When the amount of seconds specified in this parameter has passed without any activities at line level, the respective transmission or reception process (`rvscom`) is declared inactive.

12.4.1.1.2 COMTIMEOUT

`COMTIMEOUT` is the timeout for the transmission process (transmission `rvscom`).

It must be set in the `$RVSPATH/init/rdmini.dat` file.

Recommended value: `OTIMEOUT + 30` (seconds). When the transmission process fails to show any activities for the number of seconds specified in this parameter, it is regarded inactive.

12.4.1.1.3 MONTIMEOUT

`MONTIMEOUT` is the timeout for `rvs`[®] monitors.

It must be set in the `$RVSPATH/init/rdmini.dat` file.

Recommended value: 300 (seconds). When a monitor fails to show any activities for the number of seconds specified in this parameter, it is regarded inactive.

The value for this parameter should be at least twice the size of the `SLEEP` parameter value (see chapter 12.4.1.1.4) and should also exceed the delivery time for the largest file that may be expected.

Setting the `MONTIMEOUT` value to too low a value could result in a monitor failing to be able to process its activities within this interval, e.g. because it is busy delivering a large file. As a result, another monitor would stop and restart this node although it is working correctly.

Restarting an inactive monitor occurs with the `rvsrestartnode` script. This script is located in the `$RVSPATH/rvs/system` directory of each node and is executed by the monitor that detected the inactivity. Prior to starting this script all active rvs® processes on the respective node are terminated.

Example (`rvsrestartnode`):

```
#!/bin/sh
#
echo "restarting >$1<" >> $HOME/restart.log

ssh $1 ". ./profile;rvskill -q;rvsstart"
echo $1 restarted >> $HOME/restart.log
```

You can edit this script according to your needs.

All node names of the rvs® Data Center must have been configured in the `$RVSPATH/rvsenv.dat` rvs® environment file (parameter `RVSNODENAME`). This script receives the name of the failed node as a parameter (`$1`). All actions are logged in the `$HOME/restart.log` file.

You can also use `rsh` instead of `ssh`. In this case you must accordingly replace all `ssh` commands with `rsh` commands.

12.4.1.1.4 SLEEP

This parameter is not new but is explained here again for better understanding.

The value of `SLEEP` is the interval in seconds the rvs® monitor waits when no other commands are pending until it checks again for new commands to be executed.

For rvs® Data Center the rvs® monitor performs the following before it enters the sleep mode:

- rvs® Data Center monitoring: An rvs® monitor monitors all rvs® processes on the local node. At the same time it monitors the monitors of all other nodes. All activities of all monitors are logged in the database. Provided a monitor fails to write log entries to the database for a specific time, a remote monitor will detect this and cause the inactive monitor to be restarted. You can configure this time with parameter `MONTIMEOUT` (see chapter 12.4.1.1.3). The monitor also uses

the `COMTIMEOUT` parameter to verify if a transmission process is still active. A transmission process no longer active will be terminated.

- Restoring of stuck jobs: The rvs[®] monitor searches the database for stuck jobs. Any such jobs found will be restored so they can be processed in a normal way.

12.4.2 CNTMA and CNTGC

When the monitor is so busy that it cannot perform rvs[®] Data Center monitoring (because it never enters the sleep mode), parameters `CNTMA` and `CNTGC` ensure that this check takes place after a certain number of processed commands.

12.4.2.1.1 CNTMA

Parameter `CNTMA` (Count for Monitor Activities) lets you configure the number of commands a monitor has to process before it performs rvs[®] Data Center monitoring (see chapter 12.4.1.1.4).

This parameter must be set in the `$RVSPATH/init/rdmini.dat` file. Recommended value is 40.

12.4.2.1.2 CNTGC

Parameter `CNTGC` (Count for Garbage Collection) lets you configure the number of commands that must be processed before stuck jobs are being restored (see chapter 12.4.1.1.4).

This parameter must be set in the `$RVSPATH/init/rdmini.dat` file. Recommended value is 40.

12.4.3 Service Provider parameter

This chapter describes the parameters affecting the Service Provider behavior in relation to rvs[®] Data Center monitoring.

12.4.3.1.1 SPERRTO

Parameter `SPERRTO` (Service Provider Error Time Out) lets you specify the period of time in seconds following which the Service Provider regards a job as having failed.

`SPERRTO` must be set in the `$RVSPATH/rvsenv.dat` file. Recommended value: 600 (seconds). This parameter is analog to the `MONTIMEOUT` parameter for the monitor parameters.

The value of `SPERRTO` must be greater than the maximum processing time for an encryption/compression job.

12.4.3.1.2 SPTIMEOUT

Use the `SPTIMEOUT` (Service Provider Time Out) parameter to specify the time in seconds following the expiry of which the Service Provider changes to the sleep mode if there are no jobs to be processed (see also parameter `SLEEP`, `SPTIMEOUT` is analog to `SLEEP`). Before changing to the sleep mode the Service Provider performs the following actions:

- Service Provider monitoring: The Service Provider uses the values of the `SPERRTO` parameter to check if all Service Provider Workers are still active. A Service Provider Worker no longer active will be terminated.
- Restoring of stuck jobs: The Service Provider searches the Service Provider directories for jobs (job files) without any Service Provider Worker. The appropriate jobs will be restored when such jobs are found.

`SPTIMEOUT` must be set in the `$RVSPATH/rvsenv.dat` file. Recommended value: 1 (second).

Note: A Service Provider Worker is a process the rvs® Service Provider starts for each job. The Worker processes the assigned encryption/compression job and terminates. Please refer to chapter 2 of the Reference Manual for more information on the technical basis of rvs® Service Provider.

12.5 How to start and stop rvs® Data Center

The present chapter describes how to start and stop rvs® Data Center and individual nodes.

12.5.1 rvs® Data Center start

By executing the rvs® Data Center `$RVSPATH/rvs/system/farmstart` script you can start all rvs® nodes at the same time (the entire rvs® Data Center). You must edit this script to match your configuration (node names).

Example (`farmstart` for 2 nodes):

```
#!/bin/sh
#
# start rvs farm
#
NODE1=farmnode1
NODE2=farmnode2

echo ">>> starting rvs farm"

rvsstart
```

```
echo ">>> farm node started on $NODEIP"

if [ $NODEIP = $NODE1 ]; then
    ssh $NODE2 ". ./profile;rvsstart"
    echo ">>> farm node started on $NODE2"
fi
if [ $NODEIP = $NODE2 ]; then
    ssh $NODE1 ". ./profile;rvsstart"
    echo ">>> farm node started on $NODE1"
fi
```

This sample script starts rvs[®] nodes `farmnode1` and `farmnode2`. All nodes to be started must be specified in the script. In addition the `NODEIP` (node computer name) parameter must be specified in the `.profile` UNIX file. You can also use `rsh` instead of `ssh`. In this case you must accordingly replace all `ssh` commands with `rsh` commands.

12.5.2 rvs[®] Data Center stop

You can stop all nodes of an rvs[®] Data Center at the same time with this script. The script is named `farmstop` and is located in the `$RVSPATH/rvs/system` directory. You must edit this script to match your configuration (node names).

Example (`farmstop` for 2 nodes):

```
#!/bin/sh
#
# stop rvs farm
#
NODE1=farmnode1
NODE2=farmnode2

echo ">>> stopping rvs farm"

rvsstop

echo ">>> stopping farm node on $NODEIP"

if [ $NODEIP = $NODE1 ]; then
    ssh $NODE2 ". ./profile;rvsstop"
    echo ">>> stopping farm node on $NODE1"
fi
if [ $NODEIP = $NODE2 ]; then
    ssh $NODE1 ". ./profile;rvsstop"
    echo ">>> stopping farm node on $NODE2"
fi
```

This sample script stops rvs[®] nodes `farmnode1` and `farmnode2`. All nodes to be stopped must be specified in the script. In addition the `NODEIP` (node computer name) parameter must be specified in the `.profile` UNIX file. You can also use `rsh` instead of `ssh`. In this case you must accordingly replace all `ssh` commands with `rsh` commands.

12.5.3 rvs® node start

You must execute the `nodestart` script to start a single node of the rvs® Data Center. This script is located in the `$RVSPATH/rvs/system` directory and can be executed on each node.

Example (nodestart):

```
#!/bin/sh
#
# start rvs farm node
# $1 = nodename to start
#

if test $# = 0
then
    echo "usage: nodestart <nodename>"
    exit
fi

if [ $NODEIP = $1 ]; then
    rvsstart
    echo ">>> farm node started on $1"
else
    ssh $1". ./profile;rvsstart"
    echo ">>> farm node started on $1"
fi
```

This sample script is run with the node computer name in parameter `$1` and rvs® is started on this node. The same as for the `farmstart` script applies: the `NODEIP` (node computer name) parameter must be specified in the `.profile` UNIX file; `rsh` can be used instead of `ssh`.

12.5.4 rvs® node stop

You must execute the `nodestop` script to stop a single node of the rvs® Data Center. This script is located in the `$RVSPATH/rvs/system` directory and can be executed on each node.

Example (nodestop):

```
#!/bin/sh
#
# stop rvs farm node
# $1 = nodename to stop
#

if test $# = 0
then
    echo "usage: nodestop <nodename>"
    exit
fi

if [ $NODEIP = $1 ]; then
```

```
    rvsstop
    echo ">>> stopping farm node on $1"
else
    ssh $1 ". ./profile;rvsstop"
    echo ">>> stopping farm node on $1"
fi
```

This sample script is run with the node computer name in parameter \$1 and rvs[®] is stopped on this node. The same as for the `nodestart` script applies: the `NODEIP` (node computer name) parameter must be specified in the `.profile` UNIX file; `rsh` can be used instead of `ssh`.

12.6 Working with rvs[®] Data Center

To work with rvs[®] Data Center you can use the batch interface (`rvsbat`) and rvs[®] Client/Server.

12.6.1 The batch interface (`rvsbat`)

The `rvsbat` rvs[®] batch interface is used to:

- automatically send files (`SEND`),
- create resident receive entries and job starts after send attempts (`RESENTR`, `SENDJOB`),
- create users (`USER`),
- update station changes, delete and activate stations (`MODST`, `DELST`, `ACTIVATE`),
- list and edit parameters (`LISTPARM`, `SETPARM`).

`rvsbat` can be run on any rvs[®] node. Operation is identical as with rvs[®] standalone; files to be sent and scripts must be located in the central directory.

Please refer to chapter 9 of the rvs[®] portable Reference Manual for more information on the `rvsbat` commands.

12.6.2 rvs[®] Client/Server

Operation of rvs[®] Data Center is possible with `rvsbat` and also via the rvs[®] Client/Server user interface.

rvs[®] Client/Server comprises rvs[®] Middleware and rvs[®] Client. rvs[®] Middleware must run on any node while rvs[®] Client can be executed on a remote computer (e.g. Windows XP). Please read chapter 2.10.3 "rvs[®] Data Center architecture" in the reference manual for more information on the rvs[®] Data Center system components.

The following rvs[®] Client/Server functions are available for rvs[®] Data Center:

- rvs® Data Center start and stop
- file transfer
- job, station and user administration,
- resident receive entries and job starts after send attempts
- operator commands
- display, filter and save log messages from the database and statistics entries
- show the rvs® Data Center configuration.

Please refer to the rvs® Client/Server User Manual on how to use rvs® Client/Server user interface to work with rvs® Data Center. This manual also covers the functions specific for rvs® Data Center.

12.7 Low resources

The central rvs® directory and the central Oracle database must always be accessible.

Each rvs® server must always have access to the central rvs® directory via NFS (Network File System) and to the central database via SQLNET (TCP/IP). Also refer to the illustration in chapter 2.10.3 of the Reference Manual where you find more information on the system architecture.

12.7.1 Failure of central database/central directories

Parameters DBDL and DBTO control the frequency of attempts to connect to the central database and the central directory (number of connections attempts during which period of time).

DBDL and DBTO must be specified in the `$RVSPATH/rvsenv.dat` file. Recommended values: DBDL: 10 to 60 seconds; DBTO: 6 to 30 (times).

DBTO=10 and DBDL=30 means that a total of 10 checks will be made every 30 seconds to verify if the resources (database and directories) are available again.

When the central database or the central directories fail, the executable script RECERREX from the `$RVSPATH/system/` directory is launched and rvs® is stopped on the node where the low resources occurred.

Example (RECERREX):

```
#!/bin/sh
#
# (c) gedas deutschland GmbH 2005
#
# RVS command: rvs resource error exit : recerrex
#
# program can be modified by rvs user to handle errors
#
```

```
# parameter 1: error string
#
#-----#
echo "reccerrex.sh called with >$0< >$1< >$2< >$3<" >>
$HOME/resourceproblem.log
```

Parameters \$0 - \$3 represent possible reasons for the low resources. You can edit this script to suit your demands (e.g. the end user could use this script to send an email informing the person in charge of the fact that rvs[®] was stopped due to low resources).

12.8 Logging

In rvs[®] standalone the log messages were exclusively written to the \$RVSPATH/db/rlog.log file.

rvs[®] Data Center provides the option to save the log messages as follows:

- to the \$RVSPATH/db/rlog.log file only
- to the database only, or
- to both locations: log file and database.

This can be controlled with the new LOGINDB parameter. LOGINDB must be specified in the \$RVSPATH/rvsenv.dat file.

Parameter LOGINDB can have the following values:

Value	Meaning
0	Log messages are exclusively written to the rlog.log file.
1	Log messages are written to the rlog.log file and the database.
2	Log messages are exclusively written to the database.

Recommended: 1 since rvs[®] Client/Server can only evaluate the log messages when they are in the database.

The second new parameter affecting logging is LOGFORMAT. It determines the type of information that is written to the rlog.log file (the database always contains all information).

LOGFORMAT must also be specified in the `$RVSPATH/rvsenv.dat` file.

The following values are possible:

Value	Meaning
K	Context Example: [0 23] means OperatorCommand No. 23.
P	ProcessInfo (ProcessType and ProcessID) Example: (C15320); C is the ProcessType (for rvscom), 15320 is the ProcessID.
N	Node name Example: {node1}
M	Message name, message as in rvs® stand-alone. Example: <TCPIP_READY>

Recommended: 'M'. This means the `rlog.log` file contains the same information as in case of rvs® stand-alone. Other information types (such as N, P and K) are used for internal evaluation only. For the next example they are used for information only.

Example:

The following parameters have been defined in the `$RVSPATH/rvsenv.dat` file:

```
LOGINDB=1
LOGFORMAT=KPNM.
```

Accordingly, log messages are written to the `rlog.log` file and the database. As the LOGFORMAT parameter comprises all information types, all information is written to the log file (not recommended, but is used for the present example only).

A message in the `rlog.log` file could look as follows:

```
A:      2005/02/23      14:43:56      {node1}      (X35506)[S      28]
<SENDER_STARTED> SK(29) Sender to SID (XPFR) started (with
Prot TCP/IP).
```

In rvs® stand-alone the same message looks as follows:

A: 2005/02/23 14:43:56 <SENDER_STARTED> SK(29) Sender to SID (XPFRT) started (with Prot TCP/IP).

Information regarding context, ProcessID and node name is missing.

Note: For a detailed syntax description of log messages please refer to the rvs[®] "Messages and Return Codes" manual.

An assessment of log messages occurs by rvs[®] Client/Server, which can read the log messages from the database using filters (Admin -> Log Messages window) or by external applications that read the required data directly from the appropriate database tables.

The database script `$RVSPATH/system/export_lt.sh` allows log messages to be exported from the database to a file.

Example (`export_lt.sh`):

```
#!/bin/sh
ORACLELOGIN="login@ORACLE-Servername/password"
SQLSCRIPT="/home/rvsfarm/system/export_lt.sql"
sqlplus $ORACLELOGIN @export_lt.sql
```

You must specify the respective login, Oracle server name and password values for the `ORACLELOGIN` variable in the `export_lt.sh` script. You must also accordingly edit the path defined in `SQLSCRIPT`.

`export_lt.sh` requires the `export_lt.sql` script.

Example (`export_lt.sql`):

```
set termout off /* don't show display */
set heading off /* don't show column headings */
set feedback off /* don't show number of selected records */
set pagesize 0 /* don't show page headings, breaks, etc. */
set echo off /* don't display commands, when they are executed */
set linesize 200 /* length of output line */
spool lt.txt /* output file */
select trim(LT.MSGCLASS)
|| ':'
|| trim(LT.DTCREATED)
|| '{'
|| trim(LT.NODENAME)
|| '}'[
|| trim(LT.PROCESSTYP)
|| trim(LT.PROCESSID)
|| ']'[
|| trim(LT.KONTEXTTYP)
|| trim(LT.KONTEXTID)
|| trim(LT.KONTEXTSID)
|| '']
```

```
|| trim(LT.MSGTEXT)
from LT;
spool off
exit;
```

This example exports all log messages from the database to the `lt.txt` file. You are free to decide where the output file is to be written; to do so, specify the output file path in the `spool lt.txt /*output file */` line.

Example:

```
spool /home/rvsfarm/arcdire/dblogs.txt
```

```
/* output file */ comment only.
```

12.9 Parameter changes at runtime

You can use the `setparm` operator command via `rvsbat` or `rvs® Client/Server` to change certain parameters at runtime.

This applies to the following parameters:

Parameter	Description	Value range
ODTRACELVL	Odette trace level	0 -3 Default: 0
LITRACELVL	Line trace level	0-3 Default: 0
SIDTRACE	StationID for Odette/line trace	StationID Default: blank
STATISTICS	Information regarding send/receive jobs	0 -7 Default: 2
CMDDELETE	Leave processed jobs in the database or delete them.	0 -1 Default: 1 (yes)
DTCNN1-20	Time window for connection attempts in case of connection failure	Default: 00/00/00 00:01:00 to 00/00/00 00:20:00

TCPIPRCV	Number of simultaneous receive processes for TCP/IP	0-n Default: 1
MAXX25RCV	Number of simultaneous receive processes for X.25/ISDN	0-n Default: 1
OCREVAL	Odette credit value	Max: 999 bytes Default: 99
OEXBUF	Odette exchange buffer	Max. 99 999 bytes Default: 2000

Note: Please read chapter 8.1 "rvs[®] Parameters' Overview" for more details on the parameters in the table.

You must stop and start rvs[®] Data Center when you make changes to other parameters that need to be identical on all nodes, such as VFTYP.

Glossary

A

Access Method The access method describes the way by which two stations are connected.

B

Batch Interface
(rvsbat) The batch interface of rvs[®] offers user functionality for automatic background use.

C

CCP Communications Control Program

CMX Communication Method UNIX (SINIX); CMX describes the interface to the user application, in our case rvs[®].

Communication Module
(rvscom) The communication modul of the rvs[®] system connects to another station and sends or receives files.

D

Dialogue Interface
(rvsdia) The dialogue interface of rvs[®] provides interactiv user functionality.

E

EDI **Electronic Data Interchange**

EDIFACT **Electronic Data Interchange for Administration, Commerce and Transport**

EERP End-to-End-Response. ODETTE expression für die Quittung am Ende der Übertragung bei der Sendeübertragung

ETSI European Telecommunications Standardization Institute

F

FSS

Forwarding Support Service

GNU zip Algorithmus

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software. Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Jean-loup Gailly Mark Adler
jloup@gzip.org
madler@alumni.caltech.edu

If you use the zlib library in a product, we would appreciate *not* receiving lengthy legal documents to sign. The sources are provided for free but without warranty of any kind. The library has been entirely written by Jean-loup Gailly and Mark Adler; it does not include third-party code.

If you redistribute modified sources, we would appreciate that you include in the file ChangeLog history information documenting your changes.

H

I

J

K**L****M**

MasterTransmitter
(rvsxmt)

The MasterTransmitter of the rvs[®] system coordinates send and receive processes to ensure the optimal use of the net capacity.

Monitor (rvsmon)

The monitor is the main task of a rvs[®] system. It controls transmissions sent and received and initiates automatic follow up jobs if necessary.

N**O**

ODETTE

Organization for Data Exchange by
Tele Transmission in Europe

Get the complete description of OFTP
from

<http://www.odette.org/odg4/4oftp.htm>

OFTP

ODETTE File Transfer Protocol

The ODETTE File Transfer Protocol
is the definition of a file transfer
protocol by the ODETTE Group IV for
OSI Layers 4 to 7.

International Protocol used in many
business fields (Industry, Commerce,
Finance, ..).

Operator Console
(rvscns)

The operator console provides the
administrator with rvs[®] functions to
control the rvs[®] system.

OSI

Open System Interconnection

P

PDF

Portable Document Format

Protocol

To connect two different computers
they have to follow the same protocol.

	This protocol defines actions and reactions as well as the "language" spoken.
PVC	Permanent Virtual Circuit
Q	
R	
rvsmon	See Monitor
S	
Send Entry	Order to rvs [®] which file has to be sent to which station. This entry is saved in the database.
SID	rvs [®] expression for the station ID
Station	A station is a node that can be addressed within a rvs [®] network. Each station is identified by a unique station ID (SID).
SVC	Switched Virtual Circuit
T	
TNS	Transport Name Service
Transfer Component	Control program and line driver for a special access method
TSP	Transport Service Provider
U	
V	
VDA	Verband der Deutschen Automobilhersteller Adresse: Verband der Automobilindustrie e.V. (VDA) Abt. Logistik Postfach 17 05 63 60079 Frankfurt

Tel.: 069-7570-0

VDSN

Virtual Data set Name

ODETTE expression for the file name
of a file which has been transfered via
OFTP

W

X

Y

Z

Index

- 3270 Pool Record 72
- analyse problems 56
- APPC Local LU Record 68
- APPC Mode Record 70
- APPC Remote LU Record .. 70, 71, 72
- Assignment Statements 97
- AUTODIAL 35
- Basic Functional Characteristics 9
- CMDDELETE 213
- CNTGC 201
- CNTMA 201
- CODEIN 37
- CODEOUT 37
- command description 160
- Command Line Arguments 96
- communication program 160
- COMTIMEOUT 201
- Configuration
 - FSS 60
 - KOGS 57, 60, 61
 - network 30
 - rvs® 21, 26
- Configure your rvsX 50, 54
- Confirm
 - delete resident receive entry 132
 - deletion of job start after send attempt entry 142
- Confirm deletion of
 - send request 123
 - user entry 147
- Create
 - job start after send attempt entry 138
 - resident receive entry 127
 - user entry 145
- Create new Database 152
- data conversion 10
- DBDL 209
- DBTO 209
- definitions for TNS 60
- DELAY 35, 76
- Delete
 - sent request 122
 - delst 32
- Diagnostics Record 67
- Display
 - job start after send attempt entry 140
 - resident receive entry 130
 - user entry 146
- Display / Delete
 - Transmissions 111
- Display of
 - Transmissions 113, 115, 118, 119
- DTCNN1-20 213
- EDI 8, 215
- EERP_IN 37
- EERP_OUT 38
- environment parameters 88
- export_lt.sh 212
- farmstart 205
- farmstop 206
- FTP 34, 35, 36, 51, 52, 55, 57, 61, 217
- function keys 16, 17, 100
- functional characteristics 9
- Information about
 - rvs 113, 118, 148
 - rvs® 148
- Installation 25, 53, 57, 61, 67
- installation procedure 17, 23
- Invocable Transaction Program Record 73, 74
- ISDN communication controller 60
- Job start

-
- after send attempt entries 136
 - key characteristics 9
 - Limit Number of Concurrent Senders
..... 176
 - line driver 168
 - linedriver 165
 - List of
 - data sets for transmission 123
 - job start after send attempt entries 137
 - user entries 144
 - Lists of
 - stations 109
 - LITRACELVL 213
 - Local Node Record 68
 - LOGFORMAT 210
 - LOGINDB 210
 - Low resources 201, 209, 210
 - LUNAME 30, 31, 42
 - mask
 - CDRE 132
 - CDSE1 123
 - CJSS 138
 - CRRE 127
 - CUSR 145
 - DDT0 111
 - DDT1 113
 - DDTR 118
 - DELSR 122
 - DJSS 142
 - DSNL 123
 - DSPR 119
 - DSPT 115
 - DUSR 147
 - INFO 148
 - INI 101
 - JS 136
 - JSSL 137
 - RE 124
 - RE1 130
 - REL 125
 - REU 134
 - SIDL 109
 - SJSS 140
 - SND 104
 - SNDDL 109
 - SUSR 146
 - UJSS 143
 - USRL 144
 - UUSR 147
 - MasterTransmitter 217
 - MAXX25RCV 214
 - MODE 43, 70, 88, 149, 150
 - MODST 32
 - Monitor commands .. 9, 22, 27, 49, 88, 91, 92, 93, 94, 95, 96, 97, 98, 149, 150, 153, 155, 159, 160, 162, 164, 172, 175, 176, 177, 178, 217, 218
 - Monitor suspension time 95, 172
 - MONTIMEOUT 201
 - netISDN software 57
 - netISDN utilities 58
 - Network configuration 30
 - NFS 197
 - node station 10
 - nodestart 207
 - OCREVAL 36, 169, 175, 214
 - ODETTE ID 33, 36, 51, 55, 57, 61
 - ODETTE-ID 32
 - ODTRACELVL 213
 - OEXBUF 36, 169, 175, 214
 - OFTP 50
 - OFTP server 66
 - operator commands. 92, 95, 154, 155, 162, 169
 - optional services 177
 - Oracle 197, 209, 212
 - partner station.. 42, 46, 50, 51, 52, 55, 61, 66, 80, 158, 160, 176
 - Partner Station 31, 32
 - Password 36, 42, 44, 57, 73, 74
 - PASSWORD 42
 - physical network .. 8, 9, 10, 11, 30, 33, 34, 61, 67, 68, 78, 80, 81, 157, 168, 218
 - Platforms 9
 - portable version 8

PRIORITY	34, 36, 51, 55, 57, 61, 155, 156	SDLC Connection Record	69
PROFILE	41, 80	SDLC Link Record	72
PROTOCOL	35, 36, 48, 49, 51, 55, 57, 61	SDLC Link Usage Record	74
PSESSIONS	35	SECURITY	44
PSWFROM	36, 51, 55, 57, 61	Send	109, 110, 122, 123, 136, 137, 138, 140, 142, 143
PSWTO	36, 51, 55, 57, 61	Send data set	
PUNAME	44	distribution list	109
rdmini.dat	201	SEENDBLOCKS	36, 172, 175
Receive	124, 125, 127, 130, 132, 134, 188	Server farm	197
RECERREX	209	Service Provider	204, 205
Recovery	150	Session mode	43
RECVBLOCKS	36, 170, 175	SID	33, 34, 35, 36, 41, 44, 47, 48, 49, 51, 52, 55, 57, 60, 61, 63, 65, 66, 160, 161, 178, 218
Related table	30	SIDTRACE	213
Remote transaction processing program	42	SIDTYP	32, 33, 34
Resident Receive Entries	124, 125	SLEEP	202, 203, 205
Return Codes	97	Sleep mode	203
RJE FCB Record	75	SNA LU 6.2	13
rvs [®] Data Center	197	SPERRTO	201
rvs [®] database	8, 9, 21, 26, 30, 49, 88, 92, 93, 98, 150, 151, 152, 153	SPTIMEOUT	201, 205
rvs [®] Database	149	station table	30, 41, 45, 47, 50, 54, 61, 63
rvs [®] environment	18, 19, 92	Station table	22, 27
rvs [®] Network Architecture	10	stationtable	57, 63, 65, 88, 154, 158, 159, 161
rvs [®] node station	10	Stationtable	33, 34, 60
rvs [®] nodes	10, 33	STATISTICS	213
rvs [®] path	19	stop rvs [®] processes	24, 68, 91, 92, 93, 94, 149, 150, 152, 162
rvs [®] paths	19	SVC	62
rvs [®] station table	21	SYNCLEVEL	44
rvs [®] stationtable	80	Table	
rvsenv.dat	201	related	30, 158, 159, 161
rvsrestartnode	203	station	30, 33
Sample Initialization File	96	TCPIPRCV	214

TIMEOUT	45	Using non default Database	94
Timers	202	Value Parameters	97
TPNAME	42	VDSNCHAR	37
Transmission of		virtual station	32
data set	104	What rvs [®] is	8
TYPE	44, 65, 76, 79	What rvs [®] is not	8
Update		X.25 communications	62
job start after send attempt entry	143	X.25 native communication.....	61, 64, 168
resident receive entry	134	X.25 Problem Diagnosis	62
user entry	147	X.25 Routing Information	62
Usage	74, 92	XPU	44
user interface.....	9, 10	XRPV.....	42
User rights	30, 88		
USERFIELD	37		
USERID.....	42		