Entire Output Management

Concepts

Version 3.5.3

October 2025

ADABAS & NATURAL

This document applies to Entire Output Management Version 3.5.3 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: NOM-ONOMCONCEPTS-353-20251001

Table of Contents

1 About this Documentation 1
- Document Conventions 2
- Online Information and Support 2
- Data Protection 3

2 Concepts and Facilities 5
- What is Entire Output Management? 6
- How Entire Output Management Works 8
- Cleanup 14
- TCP/IP Direct Printing 17
- Processing of Binary Data 17
- Code Page Recommendations for Heterogeneous Environments 19
- Transferring Output to Entire Output Management from Remote Mainframe Nodes 20
- Transferring Output to Entire Output Management under UNIX 20
- Printing under UNIX 21
- Direct Printing from Natural 21
- Converting the Report Format 21
- Adabas Files 23
- Container Files and Active-Data File 24

1 About this Documentation

- Document Conventions 2
- Online Information and Support 2
- Data Protection 3

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at **https://documentation.softwareag.com**.

Product Training

You can find helpful product training material on our Learning Portal at **https://learn.softwareag.com**.

Tech Community

You can collaborate with Software GmbH experts on our Tech Community website at **https://techcommunity.softwareag.com**. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software GmbH news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at **https://github.com/softwareag** and **https://hub.docker.com/publishers/softwareag** and discover additional Software GmbH resources.

Product Support

Support for Software GmbH products is provided to licensed customers via our Empower Portal at **https://empower.softwareag.com**. Many services on this portal require that you have an account. If you do not yet have one, you can request it at **https://empower.softwareag.com/register**. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

2 Concepts and Facilities

- What is Entire Output Management? 6
- How Entire Output Management Works 8
- Cleanup 14
- TCP/IP Direct Printing 17
- Processing of Binary Data 17
- Code Page Recommendations for Heterogeneous Environments 19
- Transferring Output to Entire Output Management from Remote Mainframe Nodes 20
- Transferring Output to Entire Output Management under UNIX 20
- Printing under UNIX 21
- Direct Printing from Natural 21
- Converting the Report Format 21
- Adabas Files 23
- Container Files and Active-Data File 24

This document describes what Entire Output Management can do and how it works. It covers the following topics:

When you start using Entire Output Management, you need not change your present system of print-data handling all at once. You can gradually transfer the processing of your print data step by step to Entire Output Management. This allows you a smooth transition from you present print-data handling to using the full range of possibilities for the processing of print data offered by Entire Output Management.

For print data to be processed by Entire Output Management, the applications producing these data need not be changed.

Information Concerning Supported Platforms

Information in the Entire Output Management documentation applies to Entire Output Management on all supported platforms, unless otherwise indicated.

Information related to *mainframes* applies to all supported mainframe operating systems, unless otherwise indicated.

Information related to *UNIX* applies to all supported UNIX systems, unless otherwise indicated.

Information related to *Windows* applies to all supported Windows versions, unless otherwise indicated.

What is Entire Output Management?

Entire Output Management is an output management system which is used for the processing and distribution of print data. Entire Output Management can process any kind of print data in heterogeneous environments. The print data can come from a variety of data sources:

- print data produced by applications,
- data from spooling systems,
- data stored in files.

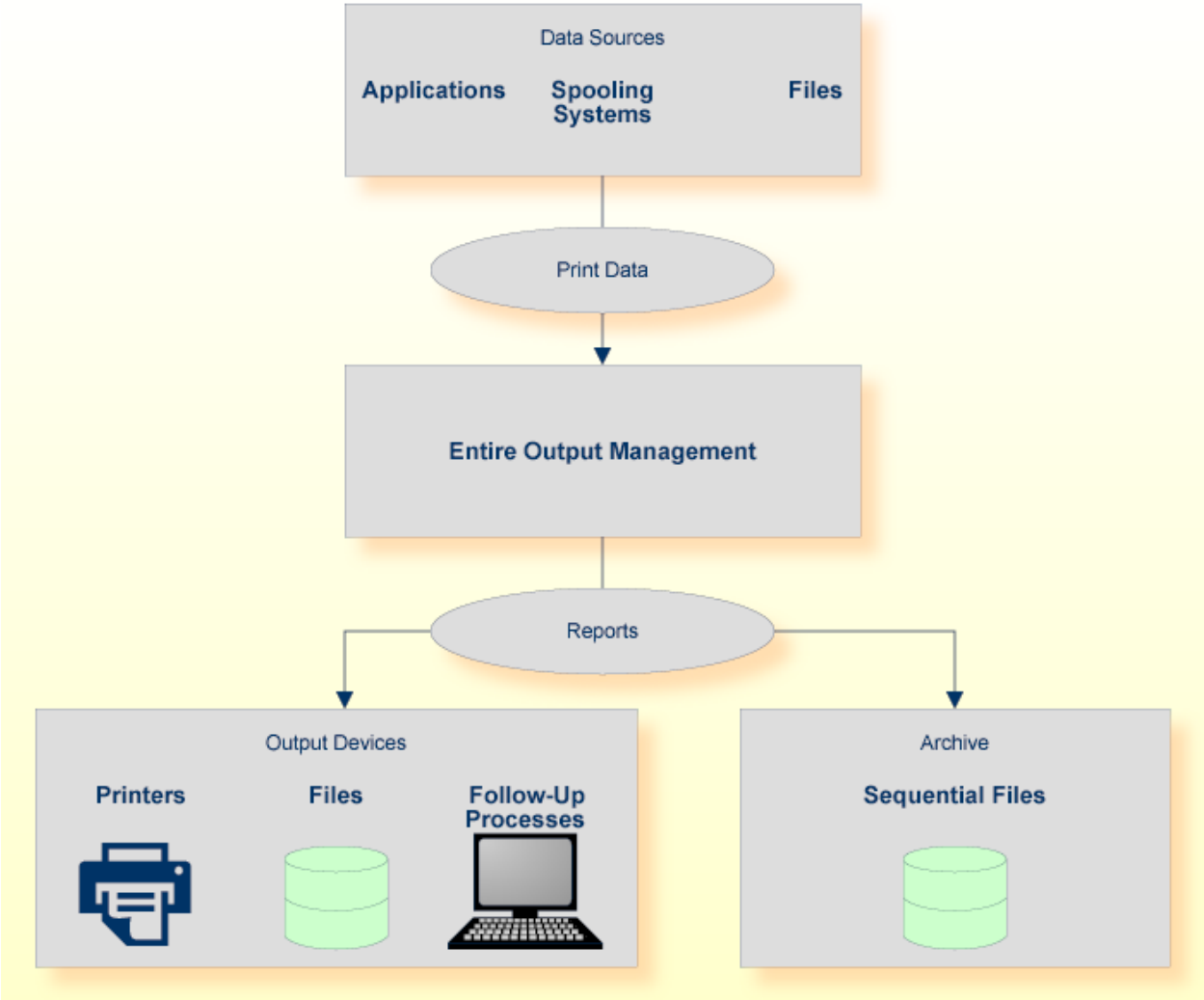
The print data actually processed by Entire Output Management are called *reports*. In the definition of a report, you select which print data from a data source are to be processed, and you determine how they are to be processed and distributed.

A report can be:

- sent to a printer for printing,
- sent to a file,
- sent to a follow-up process for further processing.

For the sake of convenience, all these "output devices" are generally referred to as *printers* within Entire Output Management.

In addition, Entire Output Management can be used to archive the print data by storing the reports in sequential files.



How Entire Output Management Works

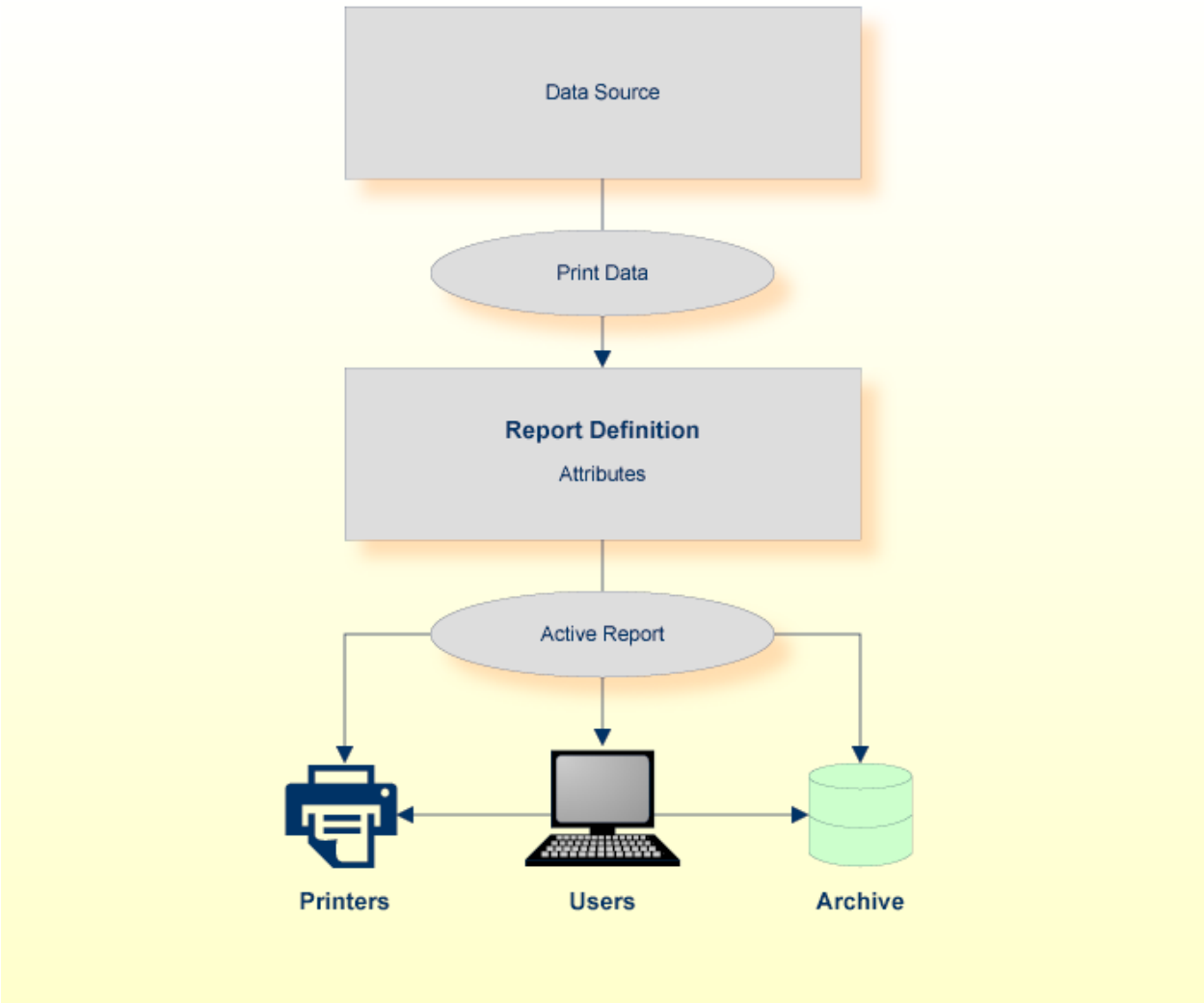
- Reports
- Bundles
- Folders
- Distribution Lists
- Printing - Logical and Physical Printers
- Naming Conventions
- Archiving and Reviving

Reports

A *report definition* consists of various *attributes*. By specifying these attributes, you determine:

- the data source of the print data to be used for the report,
- which print data are to be extracted from the data source,
- the Entire Output Management users who are to receive the report for further processing,
- how and on which printer the report is to be printed,
- if and how the report is to be archived.

The actual report which Entire Output Management creates from the report definition is called an *active report*.

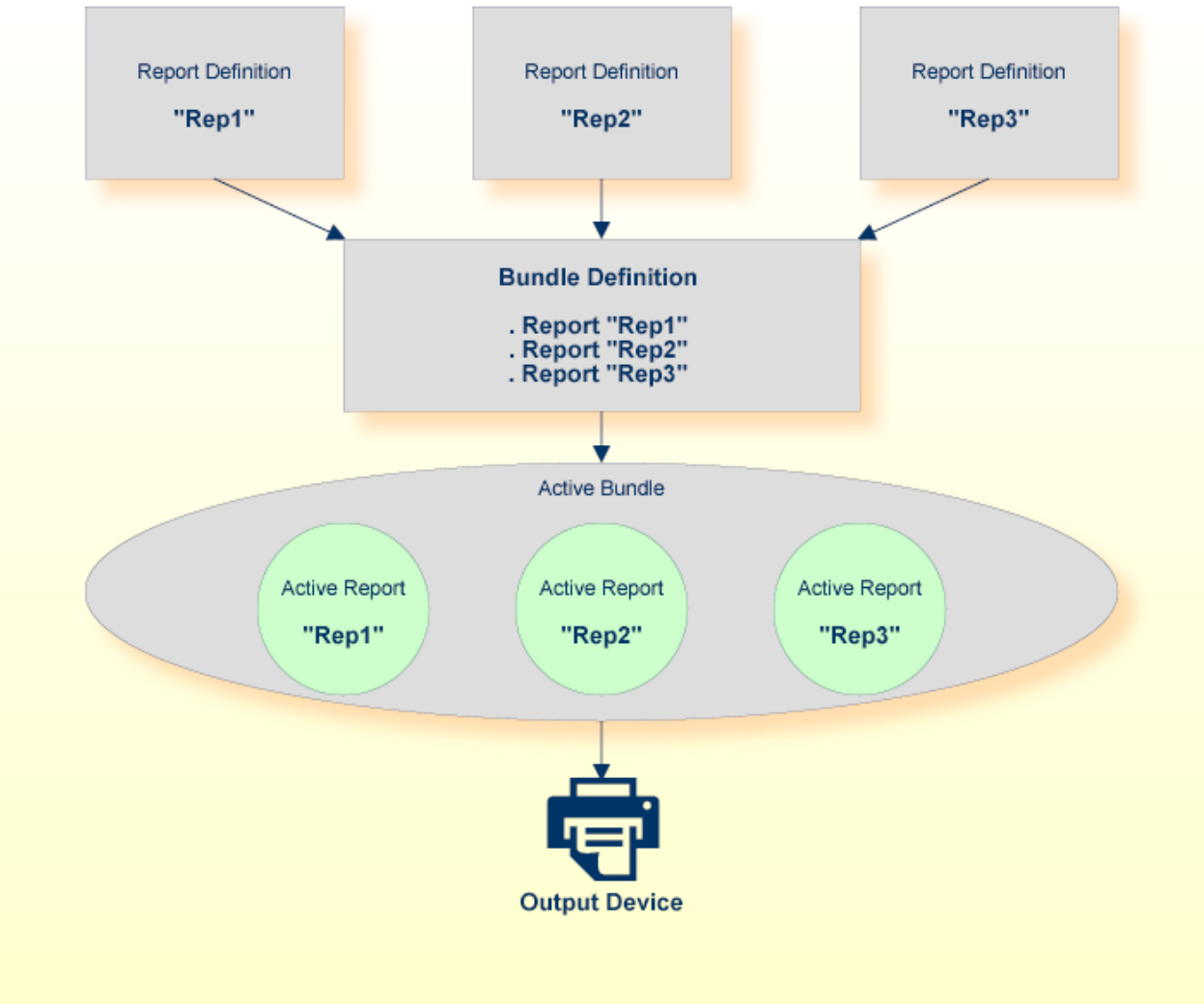


Bundles

Reports can be combined to form larger packets, which are called *bundles*. This bundling is possible even if the reports come from different data sources. The reports grouped in a bundle are processed together as a unit.

In a *bundle definition*, you specify which reports are to be part of the bundle. In addition, you can define a number of other bundle attributes to control the processing of the bundle.

When an active report assigned to a bundle is processed, Entire Output Management creates an *active bundle* based on the bundle definition.



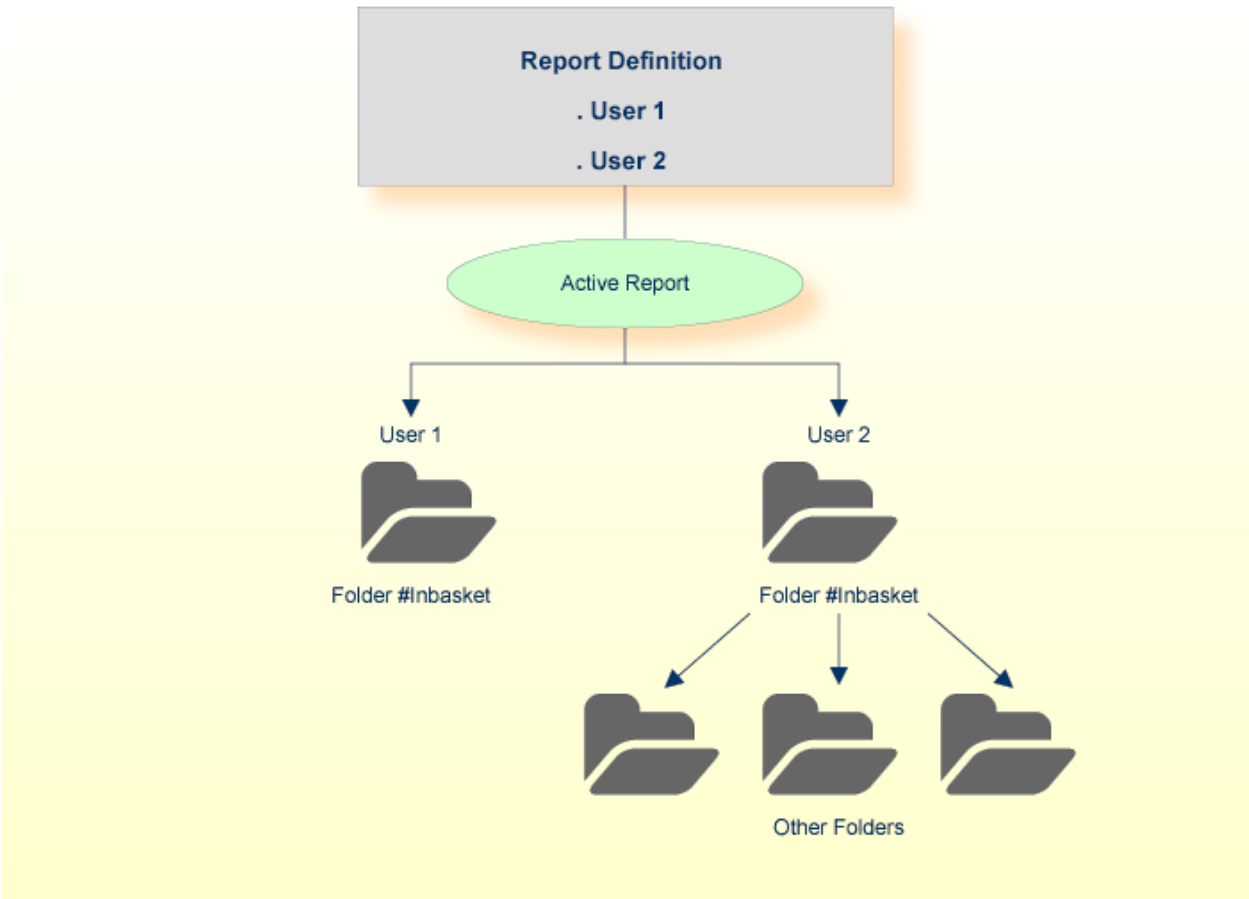
Folders

A *folder* is a container for active reports. When you define a report, you can assign to it the users who are to receive the resulting active report.

Every Entire Output Management user has a folder named `#Inbasket`. The active reports assigned to you will appear in this folder.

In addition to your `#Inbasket` folder, you can define other folders and transfer active reports from `#Inbasket` to them.

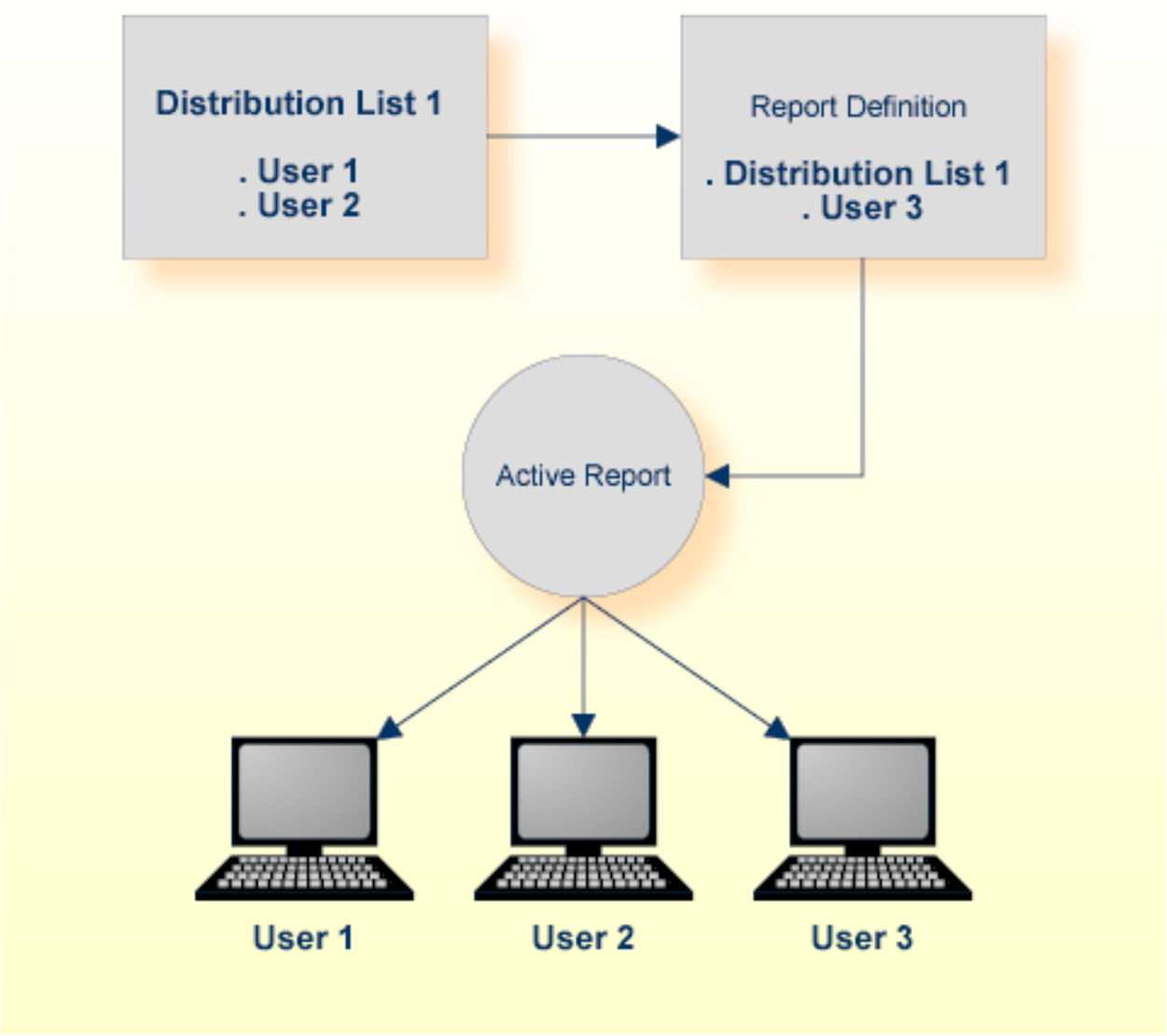
From the active reports in a folder, you can then select active reports for further processing.



In addition, you can allow other users access to one of your folders.

Distribution Lists

To make the distribution of reports to various users easier, you can create *distribution lists*. A distribution list can contain individual users, but you can also have distribution lists within distribution lists. Instead of assigning a report to multiple users, you assign it to a distribution list. It will then be distributed to all members of the distribution list.

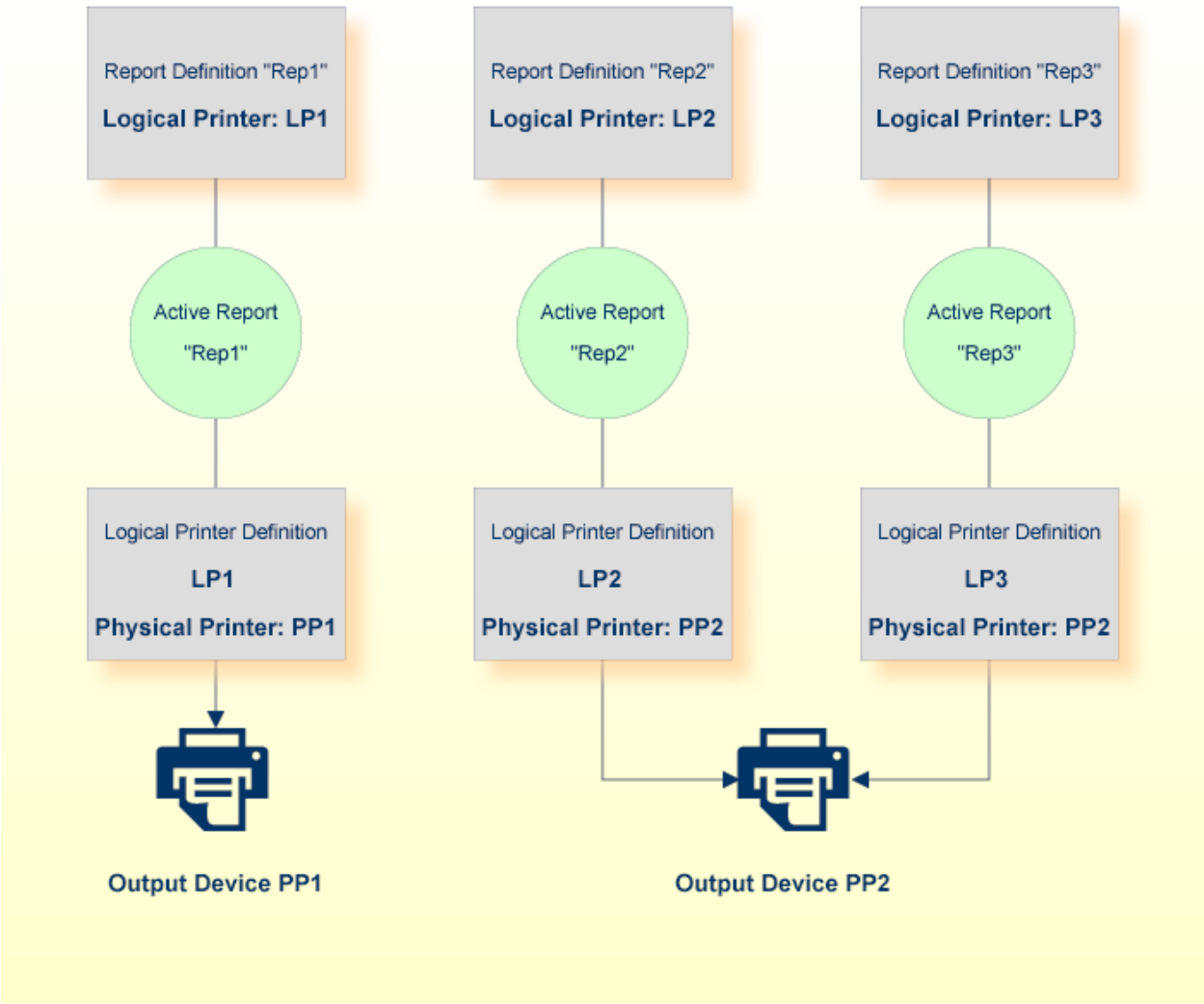


Printing - Logical and Physical Printers

The printing of an active report can be triggered either automatically depending on attributes specified in the report definition, or manually by a user.

In the report definition, you specify on which printer the active report is to be printed. A printer assigned to a report is called a *logical printer*. In the definition of the logical printer you specify a set of attributes linked to an actual *physical printer*, which determine the printing characteristics and printing format of the report on the physical printer.

If all output to be printed on a physical printer is to be printed in the same way, you only need to define one logical printer referring to the physical printer. If you wish to print different reports differently on the same physical printer, you define multiple logical printers which refer to the same physical printer, but with different printing characteristics.



Naming Conventions

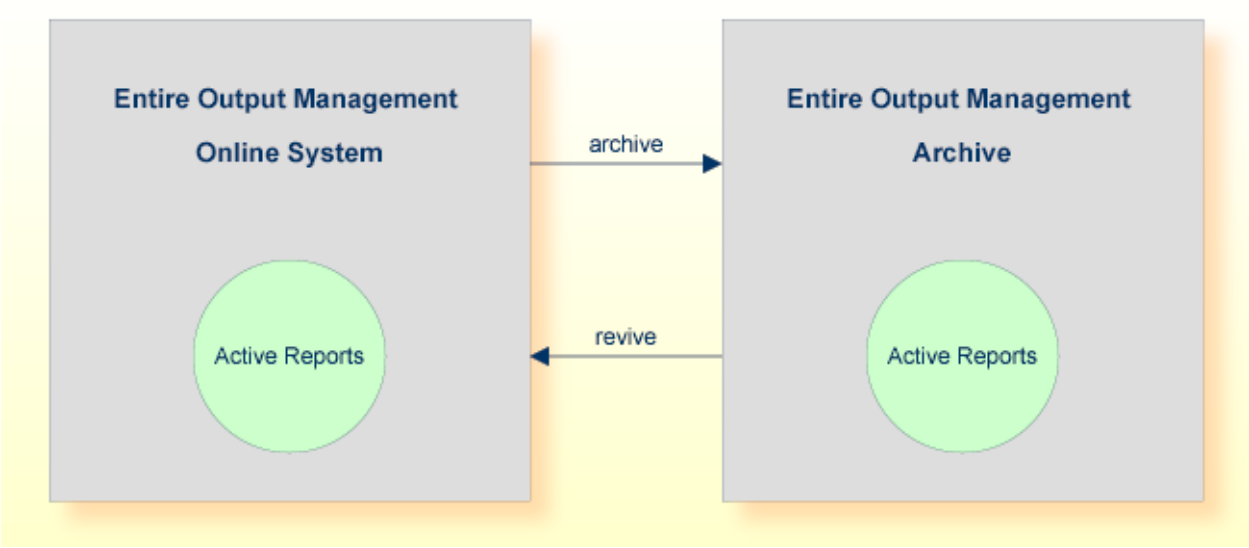
When you create and define Entire Output Management objects (reports, bundles, etc.), you are strongly advised to use object names which do *not* contain blanks or asterisks (*).

Archiving and Reviving

You can determine how long an active report is to be available online to users in Entire Output Management. You do this by defining a *retention period* in the report definition. When this period expires, the active report is longer available online.

If you wish to keep an active report, you can *archive* it on a sequential file. In the report definition, you specify how long an active report is to be kept in the archive.

If you need an active report online again, you can *revive* it; that is, you can retrieve it from the archive and copy it to the online system.



Condensing

To reduce the storage space for archive data on the sequential files to a minimum, Entire Output Management performs so-called *condense jobs*. These jobs delete from the archive all active reports whose archiving period has expired.

With the function Automatic Archiving Defaults you can define a threshold for the number of reports in an archive file that will trigger its being marked for condensation.

Cleanup

This section provides an overview of the various cleanup functions provided by Entire Output Management. It covers the following topics:

- Cleanup Types
- Daily Cleanup
- Source Cleanup
- Report Cleanup

- Daily Cleanup – Online or in Batch Mode?

Cleanup Types

There are three types of cleanup:

- daily cleanup,
- source cleanup,
- report cleanup.

The primary cleanup type is daily cleanup. Entire Output Management performs its daily cleanup once per day on its next activation after the time specified in the system defaults (option 8.1.1). If no time is specified there, the daily cleanup will be done on the first activation after midnight.

Source cleanup and report cleanup are optional and can be activated in option 8.1.7. Each can be activated individually and a schedule defined for it. They may be scheduled to run whenever you wish, using a calendar to specify on which days of the week or month they are to run, whether they should run before or after non-working days, the time window during which they are to run and how often. For example, you could define that source/report cleanup is to run every 2 hours between 08:00 and 20:00 on Mondays, Wednesdays and Fridays.

Daily Cleanup

The daily cleanup processes all expired entities:

- Active reports that have expired are deleted or marked for archiving, as appropriate.
- Active bundles, printouts and log records that have expired are deleted.
- Revived reports are “unrevived”, that is, the contents are deleted if the report was revived to the database and the active report is reset to its archived state.
- Archived active reports are deleted.
- Active report sources (spool files or container file entries) are checked and, if no longer needed, deleted.

Source Cleanup

Source cleanup checks Entire Output Management sources (for example, JES spool files in Entire Output Management’s temporary classes or entries in the container files) to see if they are still needed - and deletes them if they are not needed. They are no longer needed when there is no active report with location "S" referring to them. Container file entries which are no longer needed are deleted even if Spool Cleanup is set to "N".

Report Cleanup

Report cleanup checks all active reports with location "S" to establish if the source file is still available. If it is not, the active report is deleted.

Daily Cleanup – Online or in Batch Mode?

The daily cleanup process may entail a lot of work (typically it can run between 30 and 60 minutes in a production environment), and during this time the monitor can do nothing else; this means that no reports, bundles or printouts can be processed until the daily cleanup has finished. This is because the monitor is a single task.

To avoid this problem, the daily cleanup can run in an asynchronous batch job. To do this, you execute the program NOMCLEAN in a standard batch-mode Natural (with the necessary parameters set, such as LFILE 206). This batch job can then be activated by a job scheduler to run a couple of hours before the Entire Output Management monitor is scheduled to perform the daily cleanup. When the monitor is about to perform the daily cleanup, it can ascertain that NOMCLEAN has ran and so it will not have to do anything.

Communication

NOMCLEAN and the monitor keep each other informed of the current status by using two control records in the Entire Output Management system file. These control records are accessed using the view NRM-ARCHIVE-TASK and the descriptor M-MONITOR-ID = 'CLEANUP' or M-MONITOR-ID = SYSNOM (actually the library Entire Output Management is running from).

The communication paths are as follows:

- NOMCLEAN at its start sets SYSNOM's CLEANUP-IDENTIFIER to CLEAN, CLEANUP-ACTIVE-NOW to TRUE and CLEANUP-RESCHEDULE to FALSE to show that it is active.
- RMONITOR (monitor main loop) only deletes no longer needed sources if SYSNOM's CLEANUP-IDENTIFIER is not CLEAN or CLEANUP-ACTIVE-NOW is not TRUE (that is, only if NOMCLEAN is not currently active). In addition, to avoid concurrency problems while creating active reports, RMONITOR sets CLEANUP's REPORT-PROCESSING-NOW to TRUE while creating active reports, and to FALSE when it has finished. This is necessary to prevent NOMCLEAN from deleting reports or sources that are being worked on by the monitor.
- RMARCSCH (monitor scheduled functions) only performs daily cleanup if SYSNOM's CLEANUP-RESCHEDULE and CLEANUP-ACTIVE-NOW are both FALSE (that is, only if NOMCLEAN is not currently active and has not already run on that day). If CLEANUP-RESCHEDULE is TRUE, this means that NOMCLEAN has already run so RMARCSCH resets CLEANUP-RESCHEDULE to FALSE and updates the next scheduled daily cleanup to the same time on the next day. RMARCSCH only performs source/report cleanup if SYSNOM's CLEANUP-ACTIVE-NOW is FALSE. This is to avoid collision with NOMCLEAN which may also be cleaning up sources.
- RMSRCK (source/report cleanup) may be invoked by the monitor or NOMCLEAN. When invoked by NOMCLEAN, it needs to avoid the concurrency problem described above. It does this by

checking REPORT-PROCESSING-NOW and, if it is TRUE, uses the NPR EVENTING view to make itself dormant (repeatedly, for 5 seconds at a time) until REPORT-PROCESSING-NOW is FALSE, which is when RMSRCK resumes processing where it left off.

If multi-tasking is active (more than one monitor task is defined), the monitor cleanup functions will be performed by the main task, task 1. This will allow cleanup to be separated from other monitor functions like active report and printout creation.

TCP/IP Direct Printing

Entire Output Management provides methods for accessing printers: It is able to directly print to TCP/IP printers. This print method is intended to access printers that have either got a dedicated IP address or can be reached using printer queue names of printer servers.

This print method is intended for direct printing of shorter documents directly from online applications. However, it is executed in a Natural subtask and no online space (such as CICS memory) is affected. No spooling system is required, and no extra address space will be used, not even an ESY server. This makes this print method independent from batch jobs, Broker, CA-SPOOL, ESY, JES, and Natural Advanced Facilities.

Performance enhancements are achieved, and 4 MB of space in an Entire Output Management printer subtask per 1000 pages of text output are needed as soon as the printer (server) acknowledges the reception of the printout.

At print time, the printout will be initiated online or by the monitor and executed in one of the defined print tasks of Entire Output Management. At execution time, the printout will be kept in virtual memory and then sent to TCP/IP directly (via socket programming) using the LPR/LPD protocol (RFC1179).

Processing of Binary Data

Entire Output Management also allows you to process binary data. This means that any kind of file or printout can be kept in Entire Output Management, archived, printed, distributed, and passed to a destination system.

This is possible with UNIX and Windows file systems, but not with mainframe spool systems.

Note: If binary data are read from a mainframe source, subsequent processing may cause Natural abends like IW060 or 0C7. On mainframes, only text data can be processed. For performance reasons, however, no check is performed to ensure that the input data do not contain binary data.

To transfer binary data to Entire Output Management, you define a report that receives binary data from a UNIX or Windows directory. In the report definition, the parameter "binary reading" must be set to "B" to indicate that the file is to be processed in binary format.

The report is passed through the Entire Output Management trigger queue, similar to the NOMPUT interface. Therefore the trigger queue must be activated (Trigger Container File). The opened active report receives the type "binary" (a special CC type).

A binary document cannot be browsed with the mainframe character interface. However, the Entire Output Management GUI Client is able to receive the data, store them on the PC (please note that the whole file has to be transferred) and browse (view) them with the Windows default application of the specific file type (such as ".doc" or ".pdf").

Binary active reports can be printed with UNIXLP printers. Under UNIX, a printer type NATUNIX is available which can pass the data to the "lp" or "lpr" utility (in which case the option "-l" has to be used to ensure that the data are forwarded in a transparent mode) or to a program or script, or to a file in a UNIX directory.

Please note that a binary report that is an output of a Windows printer driver will be bound to the hardware it was created for. The decision where to print the data Entire Output Management may have received a long time ago is made at creation time. Consider to use file formats like PDF if you want to keep binary data for a long time.

Entire Output Management converts binary data to the BASE64 format. This makes it possible to pass the data across platforms and code pages, because BASE64 consists of printable characters only. However, this also means that the data are larger than the original binary data stream. When active reports are output to destinations, the BASE64 data stream is decoded to binary format again.

The output of printer drivers often result in large data streams, because a printer driver creates print pages with all graphical statements of the appropriate printer language.

In theory, separation exits can be used. However, binary data cannot be separated with standard exits, as they are not readable.

For further details and examples, see the section *Setting Up Environments for Binary Documents* in the *System Administration* documentation.

Code Page Recommendations for Heterogeneous Environments

For systems which run on multiple nodes in heterogeneous environments, please observe the following recommendations concerning the use of code pages.

Always ensure that the code page used complies with you terminal emulation and system software.

Mainframe Systems

Recommended code page for IBM systems: `IBM01140` on English systems, `IBM01140` or `IBM01141` on German systems.

Recommended code page for BS2000: `EDF03DRV`.

The code page used has to be specified with the Natural profile parameter `CP`.

Non-Mainframe Systems

Recommended code page for Windows: `WINDOWS-1252`.

Recommended code page for UNIX: `ISO-8859-15`.

These are recommended for English and German systems, as they both provide English characters, German umlauts and the Euro sign.

Define the code page used:

- in you Windows/UNIX operating system;
- in Entire System Server for UNIX in the file `npr.ini` as, e.g.: `Locale_String=ISO-8859-15`;
- in the Natural parameter module, e.g.: `CP=ISO-8859-15`;
- in the Natural RPC parameter module, e.g.: `CPRPC=ISO-8859-15,CP=ISO-8859-15`;
- in the Broker attribute file, e.g.:

```
DEFAULTS=CODEPAGE
DEFAULT_ASCII=ISO-8859-15
DEFAULTS=SERVICE
CONVERSION=SAGTCHA *ICU*
CLASS=NPR, SERVER=*, SERVICE=*, DEFERRED=NO, APPMON=NO
DEFAULTS=SERVICE
CONVERSION=SAGTRPC *ICU*
CLASS=RPC, SERVER=*, SERVICE=*, DEFERRED=NO, APPMON=NO
```

For the definition of nodes and code pages within Entire Output Management, see *Default Code Pages* and *Node Definitions* in the *System Administration* documentation.

Transferring Output to Entire Output Management from Remote Mainframe Nodes

Entire Output Management reads the data in binary form from Entire System Server, using Entire Network or Entire Systems Management Adapter (BS2000 only) as transportation layer. The data are copied into the trigger container file. Using Natural (with a `MOVE ENCODED` statement), Entire Output Management converts the data from the code page of the remote mainframe node to the code page of Entire Output Management.

For further information on code pages, see *Unicode and Code Page Support* in the Natural documentation.

Transferring Output to Entire Output Management under UNIX

Entire Output Management can handle text output from mainframe operating systems and from UNIX or Windows systems. This output has to be stored as ASCII or EBCDIC files, on mainframes with or without control characters.

Basically, Entire Output Management on UNIX handles the same output formats as on mainframes unless binary data are to be processed. Binary data have to be converted into the ASCII data format before being processed by Entire Output Management.

To get output data from UNIX or Windows directories, use the *Node Definitions* described in the *System Administration* documentation.

UNIX applications can print into a Entire Output Management file directory (ASCII files) or access spoolers like CUPS to print output to the queue of a virtual printer. In this case, the printer queue of CUPS has to be connected to a CUPS backend like "pipe" (default: `/usr/lib/cups/backend`) which stores the data as ASCII files in the Entire Output Management directory defined in the node definitions.

However, some interfaces do not exist under UNIX and can therefore not be used in an Entire Output Management UNIX environment:

- CA Spool
- NOMPUT

Printing under UNIX

Under UNIX, Entire Output Management is designed to only print on the printer types NATUNIX and DISKUNIX. They are described under *Special Printer Types* in the *System Administration* documentation.

Direct Printing from Natural

Instead of printing output from Natural applications in a spooling system, you can route the output directly to an Entire Output Management container file, without having to create any intermediate files in a file system.

This is only possible on mainframes. See *Printing from Natural to Entire Output Management Directly* in the *Installation and Customization on Mainframes* documentation.

Converting the Report Format

- Converting When Printing
- Conversion of Special Characters

Converting When Printing

It is possible to convert the output of a report to common multimedia file formats when it is printed. The printer type DISKUNIX is available to convert a file after it has been written to disk. As a prerequisite the report to be converted must be stored in Entire Output Management in the format "text" (ASCII, ASA, machine code), PDF or PostScript (or mixed in bundles). The DISKUNIX printer will convert text reports to the desired format according to the format definitions. Reports that are already in PDF or PostScript format will not be formatted because they are already rendered. PDF and PostScript formatted reports will only be converted to the desired output format without applying the Enscript parameters (see *Formatting Attributes for File-Format Conversion* under *DISKUNIX Printers* in the *System Administration* documentation). If separator pages are defined, however, they will follow the format definitions as they are text portions of the report. If bundles are to be printed DISKUNIX will collect all reports, regardless if they are in text, PDF, or PostScript format and put them into a single file of the desired output format. All text portions (text reports, separator pages) will be converted as defined in the format parameters. DISKUNIX uses the utilities Ghostscript and Enscript to perform the conversion; these haves to be installed on the destination UNIX system to which DISKUNIX writes.

IF UTF-8 is used as the file code page for reading and printing, you can use the UNIX utility Uniprint as the renderer instead of Enscript. Uniprint is part of the UNIX package "yudit" (Unicode editor), which also has to be installed on the node on which the conversion is done.

The conversion is triggered by specifying the attribute field **Output Format**.

PDF formatted reports can use a mask file for every page. This mask file is a PDF file with transparent background. This file is treated as a stamp on each page: Only the parts of the mask file which are transparent will show the original report. This means that logos or forms can be integrated into a PDF report. If the mask file contains more than one page, the corresponding pages of the original report will be overlaid.

To use the overlay function in Entire Output Management, the package "pdftk" (PDF Toolkit) has to be installed on the converting node. It is available for Windows and UNIX machines. If the field **Command** is filled, this command line can be used to further process the resulting file after the conversion For example, the utilities "ps2afp" or "pdf2afp" (contained in the IBM product Infoprint) can be used to finally create AFP-formatted files for further processing.

The above-mentioned UNIX utilities are third-party products which are *not* part of Entire Output Management; Software AG neither delivers them nor provides support for them.

The DISKUNIX printer will write text files to the UNIX file system using the UTF-8 code page. If no rendering is required (such as PDF) for a text output, but the code page is to be changed to a different one, do the following: Install the utility "iconv" in your UNIX system (if it is not installed already). Then specify the DISKUNIX printer attributes (see *DISKUNIX Printers* in the *System Administration* documentation) as shown in the following example:

Command: `iconv`
Filename: `test`
Filetype: `txt`
Opt1: `-f UTF-8`
Opt2: `-t ISO_8859-15`
Parm1: `> destination.file`

In this example, the DISKUNIX printer will create a text file `test.txt` and then calls "iconv" supplying `test.txt` as input file. This file in format `UTF-8` will be converted to the file `destination.file` in format `ISO_8859-15`. To perform further operations, you write a script which calls "iconv" accordingly.

Conversion of Special Characters

Entire Output Management transfers data from and to Entire System Server on UNIX. If the transfer is performed remotely, the code pages of the source and destination machines will differ in most cases. This means that you have to implement code-page translation using one or more middleware products. As a result, you may end up with complex environments in which special characters are still not always translated to the desired characters on the destination machine.

To avoid this problem, Entire Output Management and Entire System Server send named characters instead of the characters themselves. Entire Output Management uses the trigraph feature of Entire System Server. This means that only the ASCII 7-bit and basis EBCDIC tables are used for the conversion of characters across platforms.

Adabas Files

Entire Output Management uses the following Adabas files:

- the definition-data file (logical file number 206),
- the active-data file (logical file number 91),
- one or more container files,
- the trigger container file.

You can use the same Adabas file for both the definition-data file and the active-data file (however, the logical file specifications for both 206 and 91 are necessary).

Entire Output Management can copy print data from their original source (for example, JES spool) into a container file or into the active-data file, as described below.

For information on the trigger container file, see *Trigger Container File* in the *System Administration* documentation.

Compression

Entire Output Management stores output in arrays of 16 KB. Adabas compresses null values and trailing blanks in alphanumeric fields. Text data of output often contain blocks of blanks to present the data in the form of lists or tables. These blocks of blanks are compressed by Entire Output Management itself. This results in a more efficient use of Adabas database space and fewer databases accesses, thus improving performance.

Container Files and Active-Data File

- Using Container Files
- Using the Active-Data File
- Where to Define Container Files

Using Container Files

Entire Output Management copies report sources into a container file under any of the following circumstances:

- if Entire Output Management on UNIX is used;
- if the print data come from CA Spool, Natural Advanced Facilities or the 3GL interface (including the VTAM virtual-printer application `NOMVPRNT` with the parameter `STORE=DB`);
- in JES2 and JES3, if a spool file is processed with a `DEST` specification that matches one of the destinations defined in the **Monitor Defaults** (see *Container Files*).

Separating, browsing and printing in parts would require direct access to record ranges of the output. Access to the original print data at the source, however, may not always be possible.

When container files are used, Entire Output Management stores the complete print data in a container file before processing any report definitions. The original print data are copied as a whole. All subsequent separation processing, browsing and printing in parts requires no access to the original data source, but uses the print data in the container file. Consequently processing is also very fast.

Entire Output Management keeps the whole original output stored in the database as long as there is at least one active report with location "S" (= source) pointing to it. This could mean that the container file is filled with very large output, even though only a fraction is actually needed.

You should store print data in a container file whenever intensive separation processing (separation exits), browsing or printing in parts is necessary.

Using the Active-Data File

If you use intensive separation processing, but the resulting active reports use only a small part of the original print data, you should copy the print data into the active-data file.

The storing of print data in the active-data file is controlled by the option **Store in NOM DB** in the report definition.

If this option is set to "Y", the resulting active report will be copied from the original output stored in the container file into the active-data file (active report location = "D"). When the next cleanup

is done and there are no active reports with location "S" pointing to the original source, it will be deleted from the container file.

You should use the active-data file if you have intensive separation processing, but only small parts of the original print data are needed or if the resulting active reports have very different expiration dates.

On local z/OS systems, active reports should only be copied into the active-data file if absolutely necessary (for example, to avoid accidental loss through spool deletion), because the storing of large reports in the database creates a considerable overhead.

Be aware though, that for the lifetime of the original output in the container file, reports created with the **Store in NOM DB** set to "Y" mean that those parts of the output are actually stored again in the active-data file, whereas with location "S" Entire Output Management keeps only pointers to the container file.

Bear in mind that the active report with the highest expiration date determines the lifetime of the whole original output in the container file.

Where to Define Container Files

Container files for mainframe spooling and job-entry systems, together with the spool destination that will be copied into the associated container file, are defined in the *Monitor Defaults*.

Container files for Entire Output Management on UNIX are defined in the *Node Definitions* for a UNIX or Windows node.

Container files for CA Spool are defined in the *CA Spool Defaults*.

Container files for Natural Advanced Facilities are defined in the *Natural Advanced Facilities Defaults*.

Container files for the 3GL Interface are defined in *3GL Interface Maintenance*.