ONLINE UPDATE FETCH V2.1

User's Manual

product:	Online Update Fetch version 2.1
author:	Michael Lyons
business unit:	ITC
date:	10-03-2005 [ML] consolidated 18-10-2005 [CB] reduced size of document (8 Mb → 600 Kb) 16-11-2005 [MH] added "UTF-8 in use" parameter description 12-12-2005 [MH] updated description of UpdateSignal param. "Source"
document name:	OUF v2 1 User Manual.doc

1 Introduction to OUF

The OCLC Pica Shared Cataloguing System (CBS[I]) has an open server architecture, and communicates with client applications by means of a well-defined protocol. For example, in the OCLC Pica Local Library System (LBS), the Online Update Mechanism communicates with the CBS via the Pica3 protocol to maintain an up-to-date mirror of a library's holdings in the shared catalog.

Any third-party application that conforms to the Pica3 protocol can implement similar functionality.

For example, a client application can retrieve title data which it requires by simply requesting all of the updates from CBS on titles that have been inserted, updated or deleted by the library since a given point in time, convert this data to its own internal format, and then update its local database to reflect the changes.

However, since incorrect use of the Pica3 protocol can result in serious performance degradation of the shared catalog or even damage to the data, access to the central database for third-party applications via the Pica3 protocol must be carefully controlled.

For this reason, OCLC Pica requires that all third-party applications interfacing directly to the shared catalog via Pica3 first undergo a rigorous certification procedure.

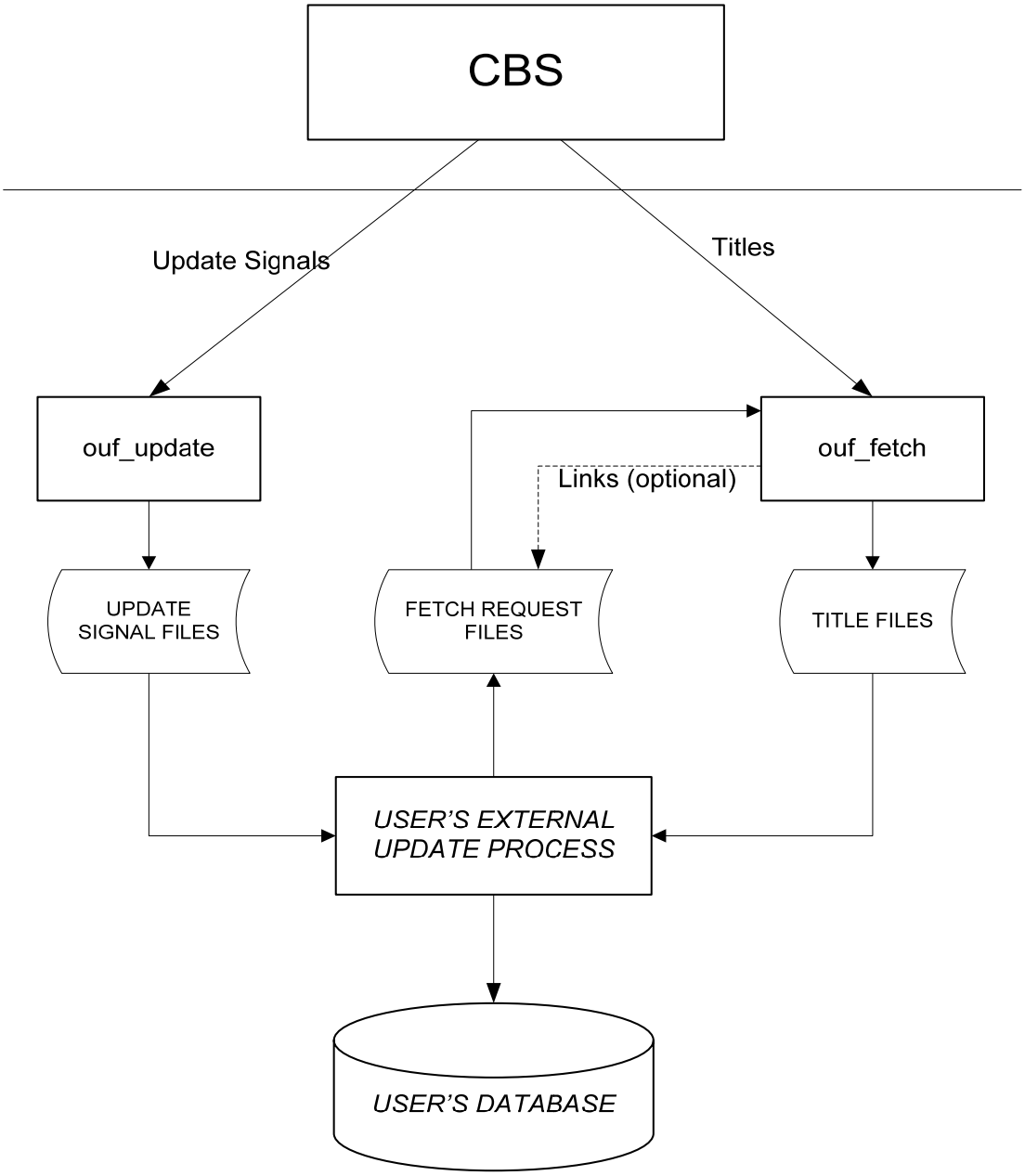
As an alternative service for customers to interfacing directly via Pica3 and the rigorous certification process, OCLC Pica offers the Online Update Fetch (OUF) product for UNIX platforms. OUF consists of a ANSI-C source-code distribution of certified production-ready applications, support shell scripts and configurations which can be easily ported to any UNIX platform.

This document contains not only detailed descriptions of the operation of OUF, but also serves as a manual of its installation, configuration and usage.

[I] CBS is an acronym for *Centraal Bibliotheek Systeem* ("Central Library System" in Dutch)

2 OUF Process Flow Diagram

The diagram below is an overview of the functional structure and data flow of OUF:



3 OUF Functions

3.1 Introduction

OUF is delivered in source code format so that users can build and, if needed, adapt the applications to meet their specific requirements.

The OUF package is based on two main applications; the title retrieval module, ouf_fetch, and the update signal retrieval module, ouf_update.

Both ouf_fetch and ouf_update are fully functional applications and are intended for use in production systems.

However, since the specific requirements for the user's local library system interface varies widely from site to site, no attempt was made to create a generic application for the user-supplied local Update Process. There is, however a test program supplied in the distribution named OUFTest which can be used to simulate a user Update Process in order to test the interfaces and data flow to and from OUF. OUFTest is described in Appendix D of this document.

In this document, the ouf_fetch and ouf_update processes are considered to be components of an integrated system, which will be collectively referred to as "OUF" unless the particular feature being described is unique to a certain module.

3.2 A Brief Overview of OUF Procedures

A typical OUF installation runs in the following fashion:

OUF uses the supplied routines in the OUF software library to establish communication with CBS using the Pica3 protocol via a TCP/IP network connection.

ouf_update retrieves update signal messages from CBS containing information about the types of updates that have been done to the titles in the database.

ouf_update creates lists of the retrieved update signals in *Update Signal Files* on the local disk.

The user-supplied external Update Process interprets these Update Signal Files and creates local *Fetch Request Files* containing title references (PPNs) to be fetched from CBS.

ouf_fetch reads the Fetch Request Files, retrieves the requested titles from the CBS database and stores them in local *Title Files*.

Finally, the user's external Update Process interprets the title information in the Title Files, and stores the titles in its local database format.

3.3 A Detailed Description of OUF Functions and Services

The next sections go into more detail about the specific functions and services available in OUF:

3.3.1 Setting Up a Pica3 Connection To CBS

- OUF runs over TCP/IP network connections.
- The network address of the destination CBS is configurable.
- A user name and password can be configured for logging in to CBS.
- The user name determines which library is selected for the titles and updates.

3.3.2 Selecting the Correct CBS Database

- The name of the database to be selected is configurable.
- The database is selected via a Pica3 command.

3.3.3 Retrieving and Processing Update Signals From CBS

- CBS maintains a log file of all modifications to the title records in the database.
- ouf_update can request records from this log file via a Pica3 command using a timestamp as starting point.
- ouf_update retrieves a group of zero to fifteen records from CBS per call.
- ouf_update then writes these interval records to an Update Signal File.
- This cycle of retrieving records from CBS and writing them to an Update Signal File continues until either no more records are available for retrieval, or until a user-configurable size or age limit for the Update Signal File has been reached.

3.3.4 Determining the Starting Date and Time For the Retrieval of Update Signals

- The timestamp of the last set of update signals retrieved from CBS is stored by OUF in its own `ouf.ini` configuration file between invocations of the system so that a newly started ouf_update update signal retriever can resume processing where the previous one left off.
- The user can override this stored timestamp value and set it to another, arbitrary point in time by simply editing it in the `ouf.ini` configuration file.

3.3.5 Producing Update Signal Files for the User-Supplied External Update Process

- When ouf_update begins to retrieve update signals from CBS, it creates a new Update Signal File in the OUF temporary directory, and begins writing the retrieved update signals sequentially to this file.
- It is possible to configure the maximum number of records in a given Update Signal File, as well as the maximum elapsed time (age) before the (non-empty) file must be closed and moved to the processing directory of the user supplied Update Process.

3.3.6 Checking For Fetch Request Files and Processing Them

- The user's Update Process reads the Update Signal Files that were created by ouf_update, and uses that data to select the titles it wishes to have retrieved.
- The Update Process creates lists of these titles in Fetch Request Files.
- A record in a Fetch Request File consists of a PPN (Pica Production Number), ILN (Internal Library Number) and a format (The ILN and format fields are both optional).
- The ouf_fetch title retrieval module processes these Fetch Request Files by reading the records from them and generating title retrieval requests to CBS.
- CBS responds to ouf_fetch with the requested title information, if it is available.
- ouf_fetch can also be configured to follow or ignore different sets of material code links in the fetched titles and to automatically fetch the titles corresponding to the links desired.
- After ouf_fetch has read and processed all of the fetch requests in a Fetch Request File and no errors have occurred, the Fetch Request File is deleted.
- If a Fetch Request File does contain syntactically incorrect requests or other errors, ouf_fetch still processes as much of the file as possible, but afterwards appends a `.err` extension to the file name to indicate the error condition, and leaves the file in the Fetch Directory for error analysis. ouf_fetch ignores files with this extension during later traverses of the Fetch Directory.

3.3.7 Checking for Retrieval Loops

If title records contain references to other title records, then loops in the retrieval of title records can occur (for example, if a cataloguer has created references in two records that refer to each other).

ouf_fetch has a simple, yet effective, heuristic for detecting and suppressing retrieval loops.

To do this, ouf_fetch maintains a history list of each PPN/ILN fetch request pair that it has seen, and prevents the same PPN/ILN record from being fetched more than a user-configurable number of times within a configurable period of time (e.g. *no more than three times within a one-hour period*). The later sections in this document describing the OUF configuration file go into more detail about configuring the loop detection options.

3.3.8 Producing Title Files For the User's External Update Process

- ouf_fetch writes the titles which it retrieves from CBS to Title Files.
- ouf_fetch sorts the retrieved titles sharing a given ILN into *ILN-specific* Title Files in the temporary directory. In other words, ouf_fetch writes a separate Title File for each ILN.
- Title File names have the default format **`title_YYYYMMDDHHMMSSCCC,`** where **`YYYYMMDDHHMMSSCCC`** is a creation timestamp for the file with an additional counter to make it possible to differentiate between files created during the same second (for example, title_20041221153620003 and title_20041221153620004).
- ouf_fetch also supports a type of macro in the configuration file which gives the user the flexibility to define his or her own naming schemes for the Title Files to fit the particular needs of the local system (e.g. **`T_024_20041221153652.6673`**, where **`24`** is the ILN of the retrieved titles in the file, and **`6673`** is the ouf_fetch UNIX process ID) *Please see Appendix B for a detailed description of this feature.*
- ouf_fetch can be configured to keep a Title File open for writing in the temporary directory until either a configurable maximum "time to live" period has elapsed, or a configurable maximum number of titles has been written to the file.
- ouf_fetch moves the Title Files from the temporary directory to the Title Directory after writing them so that the user's Update Process can access them.

3.3.9 Maintaining a Log File

The following actions are logged and timestamped:

- The opening of a network connection to CBS

- The login and logout to and from CBS
- The selection of a database
- The initial starting timestamp used for the retrieval of update signals
- The name, number of records, and the first and last timestamp contained in each Update Signal File.
- The name, number of records, deterred loops and number of processing errors encountered for each Fetch Request File.
- The name of each completed Title File and the number records it contains.
- A user-configurable level of debugging and trace information about OUF internals can also be written to the log file and/or the display screen.

3.3.10 Being Idle

- ouf_update polls periodically for update signals from CBS, and ouf_fetch polls periodically for Fetch Request Files.
- The lengths of the time intervals between both update signal polls and fetch request polls are separately configurable.
- The processes sleep (i.e. are idle) between these intervals.

4 OUF Configuration

4.1 Introduction

The layout and settings in the two main configuration files used by OUF, the FILEMAP and `ouf.ini`, are discussed in this chapter.

4.2 The FILEMAP Configuration File

A configuration file that is used by most Pica applications is FILEMAP, which is normally located in the subdirectory `$PICAMAINDIR/confdir`. It contains a number of variables that are used by OCLC Pica applications and OUF in particular.

The following table describes the main variables typically found in the FILEMAP file:

PICAMAINDIR	The directory that is used as the main directory for all OCLC Pica applications.
CONFDIR	The directory that not only contains configuration files for all applications, but also the *FILEMAP* itself. This directory should be writable by the system maintenance account only.
RUNDIR	This directory must be writable by applications and can be regarded as the current working directory of the running OUF process.
DOTINIDIR	Configuration (.ini) files should be located in the *DOTINIDIR*, which is usually the same as *CONFDIR*.
ERRORLOG	*ERRORLOG* is the pathname of the error log file of an application. Error log files contain information about errors that occurred during the execution of an application.
WARNING	The *WARNING* variable is a toggle that can be either switched to *ON* or *OFF*. If switched *ON*, then OUF will produce both error and warning messages in the error log file.
TRACE	The *TRACE* variable can be set to either *ON* or *OFF* to enable or disable tracing for OCLC Pica library functions.

Example configuration:

```
PICAMAINDIR=$HOME
CONFDIR=$PICAMAINDIR/confdir
RUNDIR=$PICAMAINDIR/rundir
DOTINIDIR=$PICAMAINDIR/confdir
ERRORLOG=$RUNDIR/errorlog_$TODAY
WARNING=OFF
TRACE=OFF
```

Although a detailed discussion of the `FILEMAP` file and its parameters falls outside of the scope of this document, it is worthwhile to note here that the `FILEMAP` can expand useful "environment" variables like `$TODAY` in the example above.

4.3 The `ouf.ini` File

OUF uses a separate file called `ouf.ini` to store its own configuration parameters.

`ouf.ini` is an editable ASCII text file which is divided into several sections or groups of parameters.

The file is arranged in a style based on the Windows .ini format.

Each group of parameters is called a *section* and is preceded by the name of the section in square brackets, followed by the settings for that section. In the following tables, each section and its possible parameters are discussed in detail[2].

Please be aware that although ouf_fetch and ouf_update are separate programs, they can, and usually do share the same configuration file.

[2] Appendix A contains an example of the contents of an entire `ouf.ini` file.

4.3.1 The [CBS] Section

The following parameters are available in the [CBS] section:

Host	*Host* is the name or IP-address of the host system providing access to CBS.
Port	*Port* is CBS port number or name to which a connection will be established.
User	*User* is the CBS user name.
Password	*Password* is the password of the CBS user.
Database	*Database is the name of the database selected to work with OUF.*
Encrypt Password	The *Encrypt Password* field can be set to *YES* or *NO*.
CBS4 in use	*CBS4 in use* should always be set to *YES* when communicating with a UNIX-based CBS. This is the default.
UTF-8 in use	*UTF-8 in use* should be set to *YES* to activate Unicode support. In that case, *CBS4 in use* must also be set to *YES*. The default value is *NO*.

Example:

```
[CBS]
Host = geronimo
Port = 1234
User = michael
Password = iamsecret
Database = BES1.SYS1
Encrypt Password = NO
CBS4 in use = YES
UTF-8 in use = YES
```

4.3.2 The [UpdateSignal] Section

The [UpdateSignal] section can contain a number of configuration parameters, timers and counters that are used by ouf_update when it is polling for update signals from CBS and for when it processes these signals.

Last Time	*Last Time* is the time of the last update signal received from CBS, and is used as the starting point the next time that ouf_update is started and begins polling for update signals. All update signals which were received since this stored last update time will be retrieved. *Note: This field (and thus* ouf.ini*) is automatically updated in place each time a new Update Signal File has been made accessible to the Update Process. This value is thus "remembered" between executions of OUF.*
Maximum Records	*Maximum Records* sets the maximum number of records that an Update Signal File may contain. When the number of records in an Update Signal File exceeds *Maximum Records,* a new Update Signal File is created, and the old file is moved to the directory where the user's Update Process can access it.
Maximum Time	*Maximum Time* sets the time limit (in minutes) until which update signals can be written to a given file. After an Update Signal File has been open for *Maximum Time* minutes or longer, a new Update Signal File is created, and the old file is moved from the temporary directory to the processing directory where the user's Update Process can access it.
Polling Interval *Polling Interval Sec*	ouf_update polls CBS for update signals at a configurable time interval. In earlier version of OUF the polling interval could only be set down to a resolution in minutes, using the variable *Polling Interval*. Later, a variable with a higher resolution of seconds named *Polling Interval Sec* was added to OUF. *One, both or neither of these The sum of the two variables Polling Interval and Polling Interval Sec is the actual value of the interval used by the application.*
Source	*Source* is the logsource of update signals to be selected. Supported values are 'H' (*high; for online updates*) and 'L' (*low; for offline/batch updates*). If this option is not set, then *all* update signals are

	processed.
Library	*Library* specifies a range of ILNs of libraries (other or more than the ILN of the user defined in the CBS section) possessing the selected records (i.e. have one or more holdings linked to these records). The values can be delimited by white space or commas, and a hyphen between two values indicates a range (e.g. 2-5 is the same as 2, 3, 4, 5). *Note: an ILN is an Internal Library Number, identifying a cataloguing library. An ILN is not the same thing as a library ID like* 1999*.*
By	*By* means "select update signals originating from this ILN only." In other words, only libraries that actually performed the transaction on the records will be selected.
Recordtype	Recordtype means "select updates for this record type only." Valid values for *Recordtype* are: *1* for titles *2* for authority records * for *any* type

Example:

```
[UpdateSignal]
Last Time = 01-11-2004
Maximum Records = 1500
Maximum Time = 75
Polling Interval = 75
Polling Interval Sec = 30
Source = H
Library =1,3-6 8
By = 3
Recordtype = *
```

In the above example, ouf_update will start polling CBS for update signals logged since November 1st, 2004, with an interval of 75 minutes and 30 seconds between each poll.

An Update Signal File produced by an ouf_update process configured as in this example will be moved to the update signal directory as soon it contains more than 1500 update signals, or when a period of 75.5 minutes has elapsed since the first retrieved update signal was written to the file.

Note that once ouf_update begins a cycle of retrieving update signals from CBS, it continues sending update signal requests to CBS until there are no more signals left to

be retrieved. It is then, and only then, that ouf_update enters the idle phase defined by *Polling Interval* and *Polling Interval Sec*.

4.3.3 The `[FetchRequest]` Section

The `[FetchRequest]` section contains a number of configuration parameters that are used by ouf_fetch when it is polling for fetch requests from the Update Process and when it is processing these requests:

Format	*Format* is the default format to be used by ouf_fetch in a retrieval command to CBS when no format was specified in a fetch request record.
Expansion	*Expansion* indicates if and how the title is to be expanded. This field should be left empty if no expansion is required. Otherwise *pica3* can be used for Pica3 expansion, or *pica+* for Pica+ expansion. *X3X* or *XPX* should be used for Pica3 or Pica+ expansion *without* MP/SP back conversion for libraries that do need this option when using OUM. *X* should be used if no expansion is desired, but the back conversion should still be skipped.
LinkTypes	The *LinkTypes* parameter can be a list of values, delimited either by commas or white space characters, and can take on case-sensitive values as used in the Pica+ tag 002@ $0, positions 1-2. The values set in the parameter list are used as pattern matching strings to compare against the linked material/record fields (link types) within a title retrieved from CBS by ouf_fetch.
Loop Check Interval	ouf_fetch detects and avoids loops in the retrieval of titles. The *Loop Check Interval* is the period in seconds used for loop checking, i.e. on a per-title basis, ouf_fetch resets the count of retrieval attempts for the given PPN/ILN pair after the *Loop Check Interval* time has elapsed.
Max PPN Retrieval Freq	The check for title retrieval loops is also based on the configurable maximum number of times a title is allowed to be retrieved within the *Loop Check Interval* period before a loop is detected. This maximum number of times is the *Max PPN Retrieval Freq*.

Polling Interval	ouf_fetch checks the fetch request directory for the existence of Fetch Request Files at a *Polling Interval* that is set in seconds.

Example:

```
[FetchRequest]
Format = P
Expansion = pica+
LinkTypes = -*f, Ga, F*
Loop Check Interval = 60
Max PPN Retrieval Freq = 3
Polling Interval = 30
```

With the settings in the example above, ouf_fetch will start checking the fetch request directory for Fetch Request Files at intervals of 30 seconds. Potential loops in the retrieval of a given title (PPN/ILN pairs) will be detected after more than 3 requests for the same title within 60 minutes.

Further requests for that same title will then be logged, but otherwise placed in an internal "exclusion list" and ignored. After an interval of 60 minutes has elapsed, ouf_fetch will release the PPN/ILN pair for this title from the exclusion list, reset the timer and counter, and allow the title corresponding to the PPN/ILN pair to be retrieved again.

The **`LinkTypes`** setting in the example above means that ouf_fetch will ignore all material type links that end with the letter 'f' and will retrieve all linked titles that have a type "Ga" or those that begin with the letter 'F' (but not "Ff"). All other links will be ignored. Appendix C gives more detail about the configuration of the **`LinkTypes`** option.

4.3.4 The `[TitleFile]` Section

The `[TitleFile]` group is used to set the maximum time in minutes that ouf_fetch will allow a given Title File to remain opened for writing in the temporary directory before it is moved to the Title directory, and/or the maximum size in bytes that a Title File in the temporary directory can reach before it must be moved.

The `[TitelFile]` section is also where the *Title Filename Pattern* option can be configured.

Maximum Time	The *Maximum Time* is the maximum length of time that a given Title File is allowed to remain opened for writing in the temporary directory before it is move out into the Title Directory.
Maximum Size	The *Maximum Size* is the maximum size in bytes that a given ILN-specific Title File is allowed to reach in the temporary directory before it is moved out into the Title Directory.
Title Filename Pattern	The *Title Filename Pattern* allows the user to specify the format of the file names that ouf_fetch gives to the Title Files that it creates.
Header Suppression Level	The *Header Suppression Level*, which can take on the values 0, 1 and 2, determines the number and type of OUF-specific headers which are placed before titles in a Title File.

Example:

```
[TitleFile]
Maximum Time = 15
Maximum Size = 250000
Header Suppression Level = 1
Title Filename Pattern = T_%I_%T.%P
```

In the above example Title Files are allowed to be open for a maximum of 15 minutes before they are moved to the Title Directory or alternatively, when the file size exceeds a maximum of 250000 bytes.

The headers preceding the titles in the Title Files generated in this example are at "Level 1," which means that the header does not contain the PPN label (i.e. the part of the header beginning with "`PPN:`"

*A full explanation of the Header Suppression Levels can be found under "*Title Files*" in Section 6 of this document, "*OUF File Types and Formats.*"*

The Title Files created by an ouf_fetch process using this configuration would have the format **`T_NNN_YYYYMMDDHHMMSSCCC.PPPPP`**, where **`NNN`** is the ILN of the titles in the file and **`PPPPP`** is the ouf_fetch UNIX process ID.

A full explanation of the syntax of Title Filename Patterns and several examples can be found in Appendix B of this document.

4.3.5 The [Trace] Section

The processing of update signals and fetch requests can be written to a trace file that can be used for diagnostic purposes.

The [Trace] section contains parameters that enable or disable tracing, set the location of the trace file, debug level, etc.

Trace	Enables/disables tracing: *ON* enable tracing. *OFF* disable tracing.
Dir	The name of the directory where the trace file will be created
Tracefile	The name of the trace file. The default name is of the form **trace_DDMMYYYYHHMM**
Replace	Determines whether a new trace file will replace (overwrite) an existing trace file, or simply append trace data to the existing trace file. *Note that this option will not have any effect when a default trace file name is used, since that filename will change and is unique between invocations of ouf_update.* *YES* replace an existing trace file *NO* append to an existing trace file
Screen	*Screen* Enables tracing to the display screen. Two values can be used: *OFF* disable screen output. *ON* enable screen output.
Debug	Enables/disable tracing *ON* enable debug messages at the default debug level. *OFF* suppress all debug messages A numerical debug level from 1-99 is also a valid parameter for *Debug;* setting this valued higher will increase the number and detail of the debug messages.

Example:

```
[Trace]
```

```
Tracefile = ouf_trace
Dir=/home/ouf/trace
Replace = YES
Screen = OFF
Trace = ON
Debug = 5
```

In this example, tracing is enabled and trace data will be logged to a trace file named `ouf_trace` written into the directory `/home/ouf/trace`.

Note that if the name of the trace file has not been configured, then OUF will generate a default name based on the clock time at the moment the file was created, in the format **`trace_YYYYMMDDHHMMSSSSCCC`**.[3]

Also in the above example, trace information will be written to a disk file only, since `Screen` is set to `OFF`.

Even if `Screen` were set to `ON` it would still be required that `Trace` be set to `ON` as well for trace output to be written to the screen.

Debugging has been turned on, and all debug messages with a severity level of 5 or less will be printed.

See Appendix B for an example excerpt of an OUF trace file.

4.3.6 The `[Directories]` Section

OUF makes use of a number of special directories:

4.3.6.1 Temp Dir	*Temp Dir* is the directory where ouf_fetch or ouf_update store their temporary files while processing update signals and fetch requests.
Update Signal Dir	*Update Signal Dir* is the directory where ouf_update stores the completed Update Signal Files for retrieval by the user's Update Process.
Fetch Request Dir	*Fetch Request Dir* is where the user-supplied external Update Process stores the Fetch Request Files for retrieval by ouf_fetch. In other words, ouf_fetch expects to find Fetch Request Files in this directory.

[3] For example, the trace file **`trace_2004102112574100`** was created by OUF on October 21st, 2004 at 12:57:41.00

Title Dir	*Title Dir* is where ouf_fetch stores its completed Title Files for retrieval by the user's Update Process.

Example:

```
[Directories]
Temp Dir = /home/ouf/temp
Update Signal Dir = /home/ouf/updsig
Fetch Request Dir = /home/ouf/fetch
Title Dir = /home/ouf/data
```

5 OUF Operation

5.1 Starting OUF

OUF can, in principle, run continuously as a background process, so one way to start the OUF processes from a UNIX shell is shown below:

```
$ ouf_fetch 2>>f_errorlog.fatal &
```

```
$ ouf_update 2>>u_errorlog.fatal &
```

The above commands will start ouf_fetch and ouf_update in the background, and redirect their standard error output to the error log files f_errlog.fatal and u_errlog.fatal, respectively.

Alternatively, one can use the supplied shell script `start_ouf`, which will start up both OUF processes in the background.

```
$ start_ouf
```

Use of the `start_ouf` script has the advantage that it sets up the shell environment so that the supplied `stop_ouf` script can later be used to automatically find and terminate the OUF processes later.

`start_ouf` also sets an environment variable which is checked to prevent multiple instances of the OUF processes from being started, and avoid the resulting contention for resources.

In either case, unless an error occurs, both ouf_fetch and ouf_update are started as detached background processes.

A table of all of the possible startup error messages follows:

5.1.1.1 Error Message Text	Explanation / Solution
5.1.1.2 Fetch request directory not found	The fetch request directory was not found. Check the directory name contained in the `Fetch Request Dir` field of the **`[Directories]`** section in `ouf.ini`. Since OUF tries to create the fetch request directory when it is not present, this error actually means that the creation of the directory failed.[4]

[4] An anomalous case is when a non-directory file such a normal file, socket, etc. already exists with the same name as configured in *Fetch Request Dir.* A "directory not found" error will be returned in this case as well.

FILEMAP not present	The *FILEMAP* file was not found. This essential configuration file contains settings for Pica applications and is normally located in the subdirectory `$PICAMAINDIR/confdir` within the home directory.
Profile environment variable not set	The environment variable `DOTINIDIR` was not found in the `FILEMAP` file. This variable contains the name of the directory where the main OUF configuration file (`ouf.ini`) is located, and is normally set to `$PICAMAINDIR/confdir`.
Profile not opened	The `ouf.ini` file could not opened, most likely because the file did not exist.
Temporary directory is write-protected	The temporary directory is not writable by OUF. OUF uses the temporary directory to create files and store data while processing Fetch Request Files and creating Update Signal Files, so this directory must be writable by the OUF processes.
Temporary directory not found	The temporary directory was not found. Check the directory name contained in the `Temp Dir` field of the `[Directories]` section in `ouf.ini`. Since OUF tries to create the temporary directory if it doesn't exist, this message actually means that the creation of the directory failed.
Title directory is write-protected	ouf_fetch uses the Title Directory to store completed Title Files, so this directory must be writable by ouf_fetch.
Title directory not found	The Title Directory was not found. Check the directory name contained in the `Title Dir` field of the `[Directories]` section in `ouf.ini`. Again, since OUF tries to create the Title directory if it doesn't exist, this

	message actually means that the creation of the directory failed.
Trace file was not created	The trace file was not created, most likely because the directory in which the trace file should be written is not writable or does not exist. Check the directory name contained in the `Dir` field of the `[Trace]` section in `ouf.ini` and also check the access permission modes of the directory.
Trace file was not opened	The trace file could not be opened, also most likely because the directory in which a previous trace file is stored is not writable or does not exist. Check the directory name contained in the `Dir` field of the `[Trace]` section in `ouf.ini` and also check the access permission modes of this directory.
Update signal directory is write-protected	ouf_update uses the Update Signal Directory to store update signals files after processing update signals, so the directory must be made writable by ouf_update
Update signal directory not found	The Update Signal Directory was not found. Check the directory name contained in the `Update Signal Dir` field of the **`[Directories]`** section in `ouf.ini`. Since OUF tries to create the update signal directory if it doesn't exist, this message actually means that the creation of the directory failed.

If either ouf_update or ouf_fetch fails to establish a connection to CBS, a P3 library error is displayed and the process is stopped. The following table contains the P3 library errors that may occur when opening an OUF connection:

5.1.1.2.1.1 Error Type	Cause / Solution

5.1.1.2.1.2 P3E_DISCONNECT	An OUF process failed to connect to CBS because of an invalid host name/IP address or because of an invalid IP port number. Check the `Host` and `Port` fields of the **`[CBS]`** section in `ouf.ini`.
P3E_IDPASSWRONG	An OUF process succeeded in connecting to CBS but failed to log on because of an invalid user name or password. Check the `User`, `Password` and `Encrypt Password` fields of the **`[CBS]`** section in `ouf.ini`.
P3E_AUTHORITY	An OUF process failed to select the database because the user has no authority to access that particular database. Check the `Database` field of the **`[CBS]`** section in `ouf.ini`.
P3E_PROTOCOL	An OUF Process failed to select the database because of an invalid database name. Check the `Database` field of the **`[CBS]`** section in `ouf.ini`.

5.2 Stopping OUF

Once the OUF processes have been started, they run continually until either their connections with the CBS server fail or until they are stopped manually.

If the `start_ouf` script was used to start the processes, as was recommended, then its counterpart script `stop_ouf` may be used to cleanly terminate the processes:

```
$ stop_ouf
ouf_fetch:  caught signal 15, cleaning up and exiting…
ouf_update: caught signal 15, cleaning up and exiting…
$
```

A process will display a message when terminated that indicates if an execution error occurred. If an error did occur, then it will be displayed on the next line. In general, errors occur only during startup, e.g. when a process fails to connect to CBS or when a particular directory cannot be created or accessed.

Also, as far as possible, the OUF processes will try at shutdown to close all of their temporary Title, Link Fetch Request and Update Signal Files and move them to their respective processing directories.

Terminating processes that were started without the `start_ouf` script requires finding out the process ID of the running OUF processes (normally via use of the ps command), and sending them **SIGTERM** or **SIGINT** signals (e.g. via the `kill` command).

5.3 Using cron to keep OUF running

As was mentioned earlier in this chapter, the OUF programs should normally run continuously as daemon processes.

However, as was pointed out earlier, an OUF process will terminate if the network connection to the CBS host fails during operation.

Since the process will not restart automatically without manual intervention, the UNIX clock daemon service, cron, can be used to periodically restart a terminated OUF process.

In the following example, a crontab file named `crontab.ouf` is submitted to the cron daemon.

This crontab will cause cron to execute the `start_ouf` script at the beginning of each whole hour:

```
$ cat crontab.ouf
0 0-23 * * * /home/ouf/pica/bin/start_ouf
$ crontab crontab.ouf
$ crontab -l
0 0-23 * * * /home/ouf/pica/bin/start_ouf
$ ps -fu ouf
     UID   PID  PPID  C    STIME TTY      TIME COMD
     ouf 24773   169 80 11:41:48 ?        0:06 /usr/openwin/bin/xterm
     ouf 24776 24773 80 11:41:50 pts/2    0:03 bash -login
     ouf 28186     1 18 15:18:02 ?        0:00 ouf
     ouf 28190 28166 21 15:18:13 pts/1    0:00 ps -fu ouf
```

Please note that once the `crontab.ouf` file has been submitted to the cron daemon, the `stop_ouf` script will no longer terminate OUF permanently.

The processes will be stopped by `stop_ouf`, but cron will restart them again at the start of the next whole hour.

Since the start_ouf script has checks to prevent multiple process from being started, this script will not interfere with the operation of running OUF processes.

In order to stop cron from restarting OUF, the following command must be issued:

```
$ crontab -r
```

This command will remove the user's crontab from the cron's queue and will thus stop `start_ouf` from being executed each hour.

The OUF processes themselves can now be stopped by means of the `stop_ouf` script:

```
$ stop_ouf
```

5.4 Run-time errors

Most run-time errors will not lead to the termination of OUF, but will be written to an error log file called `errorlog_`*`YYMMDD`* in the `$PICAMAINDIR/rundir` subdirectory. An example of the error log file is shown below.

```
---
TIM : 2004-10-11 14:24:28
PID : 22654
NAM : ouf
ERR : OUF
LOC : OUFApplication
COM : OUF: Unknown Pica3 format in line 2 of file
'/home/ouf/fetch/fetch.lst'.
---
```

In this example, the error occurred on October 11th, 2004 at 14:24:28. The ouf_fetch process ID at that time was 22654, and the error was caused by a format error in a Fetch Request File.

The following table contains an overview of run-time errors that may occur:

Run-time Error	Cause
Unknown Pica3 format in line n *of file* x.	ouf_fetch encountered an invalid Pica3 format in line ***n*** of the Fetch Request File named ***x***. Processing of the file continues, but the file is renamed after processing by the addition of the .err extension.
Syntax error in line n *of file* x.	ouf_fetch encountered a syntax error in line ***n*** of the Fetch Request File called ***x***. Processing of the file continues, but the file is renamed after processing by the addition of the .err extension.
Profile not updated	This error occurs if ouf_update cannot update its profile, most likely because the profile could not be found at its expected location in the `confdir` subdirectory or the permission modes of the directory or the configuration file itself have been set so they are not writable by ouf_update.

As pointed out earlier, there is one fatal error that can lead to the termination of the OUF processes.

If the network connection between an OUF process and the CBS host is lost, then the process will be unable to perform its primary function and will therefore terminate, closing all currently open temporary files and make them accessible to the user's Update Process. The error ***P3 Library error (P3E_DISCONNECT)*** will be written to the error log file.

6 OUF File Types and Formats

Note: Before the OUF processes run a number of subdirectories must be present. See Configuration: [Directories] for more information on this subject.

6.1 Naming Conventions

OUF uses the following default file naming conventions:

updsig_<*timestamp*> for Update Signal Files;

title_<*timestamp*> for Title Files.

The <*timestamp*> part should be filled with the date and time of the file's creation and a counter in the format: YYYYMMDDHHMMSSCCC.

OUF supports user-configurable Title File names, as described in Appendix B of this document.

The names of the Fetch Request Files have no set format, except for the fact they *may not* end with the extension '.err' which ouf_fetch uses to rename erroneous Fetch Request Files in the Fetch Directory so that it can later recognize and skip these files during a subsequent Fetch Directory traversal.

6.2 Update Signal Files

Update Signal Files are plain ASCII text files. Each line contains an update signal record in the following format (an underscore denotes white space):

```
<iln>_<library>_<date>_<time>_<type>_<ppn>_<lvl>_<copy>_<mat>[_<epn>]

 <iln>       = iln of the user that made the update
 <library>   = library of the user that made the update
 <type>      = type of update
               [IN]  = insert
               [DE]  = delete
               [MU]  = update
               [OU]  = out-of-use (remove all copies)
 <ppn>       = Production number of title
 <lvl>       = title level
               0 = shared
               1 = local
               2 = copy
 <copy>      = copy sequence number (01..99) or –
 <mat>       = Type of material (XX)
               T*    = authority-record
               ..    = title
 <epn>       = production number of copy
```

The following Update Signal File was generated by ouf_update with the *Last Time* field set to 01-09-1996:

```
ILN Library    dd-mm-jjjj hh:mm:ss.ttt.ttt type PPN       l ex ma EPN

001 0005       06-09-1996 13:45:51.357.318 [MU] 090081811 0 00 Tp
001 0005       20-09-1996 11:41:06.514.331 [MU] 06928993X 0 00 Tp
001 0005       30-09-1996 15:16:37.132.068 [IN] 854312471 2 01 000642053
001 0005       30-09-1996 15:16:37.459.577 [MU] 854312471 0 00 Aa
001 0005       30-09-1996 16:02:57.391.042 [IN] 854323392 2 01    000642061
001 0005       30-09-1996 16:02:57.654.483 [MU] 854323392 0 00 Aa
001 0005       30-09-1996 16:04:20.654.211 [MU] 854312463 2 02    000070084
```

Remarks:

There are three levels of title records in the central system: global, local and copy records. Each update of a record is separately marked in the list of updates. Thus, for an insert of a title record with three copy records, there will be four records in the list of database updates.

Since the central system is concurrent and multi-user, there is no guarantee that the records in the list of updates for one title will not interspersed with update records of other titles.

The transfer of a copy record from one title (ppn-1) to another title (ppn-2) is registered with two update records in the list: 1 delete record for ppn-1 and one insert record for ppn-2, both with the same epn.

6.3 Fetch Request Files

Fetch Request Files are standard ASCII files. Each line contains a fetch request record in the following format (_ denotes a blank):

```
<ppn>[_<iln>][_<format>]
 <ppn>       =    Production number of title
 <iln>       =    iln for which to fetch local data
 <format>    =    requested format
                  P = Pica+
                  D = Pica3
```

The following is an example of the contents of a Fetch Request File:

```
854312463 11 P
854323392 11 P
854312471 11 P
06928993X 11 P
090081811 11 P
```

6.4 Title Files

Title Files are not plain text files, but contain ASCII text and marker characters.

A Title File has the following structure:

```
Title file      =   {<title record>}
<title record>  =   <header>[<body>]
[header]        =   PPN:<ppn>_<format>[_<result_code>_<message>]<R>
<ppn>           =   production number of title
<format>        =   requested format
                    P = Pica+
                    D = Pica3

<result_code>   =   a number denoting an error

<message>       =   the error message in plain text. If an error is
                    present the record body is omitted.
<body>          =   {<dbslvl><length><time><R><title_component><R>}
<dbslvl>        =   database level
                    BIB = shared
                    LOK = local
                    Eii = copy

<length>        =   length of <title_component>, four digits
<time>          =   time last update of <title_component>\
                    format: hh.mm.ss.hhh
<title_compo-   =   {<title_tag><R>}
nent>
<title_tag>     =   title tag, formatted in either Pica+ or Pica3
<R>             =   Carriage return character (\r)
```

An excerpt of a Title File created by ouf_fetch is shown below, in side-by-side hexadecimal and printable ASCII dump output.

```
Offset Hexadecimal output                  ASCII output
0000:  50 50 4e 3a 38 35 34 33 32 33 33 39 |PPN:85432339
0012:  32 20 50 0d 42 49 42 30 32 32 39 30 |2 P.BIB02290
0024:  30 2e 30 30 2e 30 30 2e 30 30 30 0d |0.00.00.000.
0036:  30 30 31 41 20 9f 30 31 30 31 37 3a |001A .01017:
0048:  32 31 2d 30 37 2d 38 37 0d 30 30 31 |21-07-87.001
0060:  42 20 9f 30 30 30 30 30 3a 30 30 2d |B .00000:00-
0072:  30 30 2d 30 30 9f 74 30 30 2e 30 30 |00-00.t00.00
0084:  2e 30 30 2e 30 30 30 0d 30 30 31 44 |.00.000.001D
0096:  20 9f 30 39 39 39 39 3a 39 39 2d 39 | .09999:99-9
0108:  39 2d 39 39 0d 30 30 32 40 20 9f 30 |9-99.002@ .0
0120:  41 61 78 0d 30 30 33 40 20 9f 30 30 |Aax.003@ .00
0132:  33 30 30 32 30 34 33 33 0d 30 31 30 |30020433.010
…
```

The *Header Suppression Level* configuration option setting can be used to allow or suppress the detail in the header preceding the actual title data.

A Header Suppression Level of 0 means *no* suppression, and would produce output similar to the example above, where the header beginning with "PPN:" is present:

```
PPN:854323392 P.BIB022900.00.00.000.001A.01017:
```

A Header Suppression Level setting of 1 would suppress the "PPN:" header, so that we would then see:

```
BIB022900.00.00.000.001A.01017
```

Setting the Header Suppression Level to 2 would also cause the "BIB:…" header to be stripped out as well.

Appendix A: *An Example Of an `ouf.ini` File*

```
[CBS]
Host = develop.pica.nl
Port = 1140
User = michael
Password = 1961
Database = BES1.SYS1
Encrypt Password = NO
CBS4 in use = YES
UTF-8 in use = YES

[TitleFile]
Maximum Size = 100000
Maximum Time = 5
Title Filename Pattern = T_%I_%T

[FetchLinks]
Maximum Records = 4000
Maximum Time = 3600
LinkTypes = -T*,**

[FetchRequest]
Format = P
Expansion = pica+
Loop Check Interval = 4000
Max PPN Retrieval Freq = 3
Polling Interval = 60

[Trace]
Tracefile = fetch_trace
Dir = /users/michael/pica/ouf/trace
Replace = NO
Screen = ON
Trace = ON
Buffer = OFF
Debug = 11

[Directories]
Temp Dir = /users/michael/pica/ouf/tmp
Fetch Request Dir = /users/michael/pica/ouf/f_d
Title Dir = /users/michael/pica/ouf/t_d
Update Signal Dir = /users/michael/pica/ouf/u_d

[UpdateSignal]
Maximum Records = 750
Maximum Time = 1
; 2 minutes
Polling Interval Min = 0
Polling Interval Sec = 30
Recordtype = *
```

```
Library = 15  9,6 13
Last Time = 13-01-2005 16:34:33.480000
```

Appendix B: *Title Filename Patterns*

The **`Title Filename Pattern`** option in the **`[TitleFile]`** section of the configuration enables the user to specify simple and more complex naming schemes for the Title Files written by ouf_fetch.

A pattern string is allowed to consist of runs of the alphanumerical characters a-z, A-Z and 0-9, and the hyphen, underscore and period characters.

In addition, three special tags are recognized in a pattern and are replaced by ouf_fetch during the generation of the filenames:

Pattern	Effect
%T	*%T* will be replaced by a timestamp based on the current UNIX local clock time on the machine where ouf_fetch is running. The timestamp will have the form **`YYYYMMDDHHMMSSSSCCC`**, where CCC is an additional counter to ensure uniqueness the generated file names of files created during the same clock second.
%I	*%I* will be replaced by a four-digit (with leading zeroes) representation the ILN of the Title File when ouf_fetch knows the ILN of an ILN-specific Title File. If there is no specific ILN for the Title File, then the default ILN of *0000* will be used.
%P	*%P* will be replaced by the five-digit representation of the UNIX process ID of the current ouf_fetch process.

All of the above patterns may appear multiple times in the Title Filename Pattern

A few examples should clarify the Title Filename Pattern syntax:

Assume that the current ouf_fetch process ID is 8681 and the ILN is 23 for these examples

Pattern	**Generated file name**	**Comments**
`title_%T`	`title_17012005180234001`	This is actually the default format for Title Files, and is the same result as when the Title Filename Pattern is not

		specified at all
T_%I_%T.%P	T_0023_17012005180234001.08681	This may be the most useful format for ILN-specific Title File names.
hello	hello_17012005180234001	When a pattern does not contain a %T tag, then ouf_fetch will automatically append a timestamp, preceded by an underscore, to the generated names, in order to avoid naming conflicts.
%T.%P	17012005180234001.08681	
%I%I%I	002300230023_17012005180234001	
T$!*″″?	T_17012005180234001	"Illegal" characters in a pattern are simply ignored. Since there was no %T tag in this example, a timestamp was automatically appended.

Appendix C: *Links and the LinkTypes Configuration Option*

Introduction to Links

A titles retrieved from CBS by ouf_fetch can contain up to *** so-called Links to other titles. A Link consists of a two-character type identifier, followed by the PPN of the linked title

The following explains how it is now possible to configure ouf_fetch to follow a user-configurable subset of such links and have ouf_fetch retrieve the linked titles automatically.

The LinkTypes Configuration Option

Links are typed via two-character case-sensitive codes representing, for example, the material type. Some examples are "Tp," "Ag," and "Hp"

Since the default behavior of ouf_fetch is to ignore all links completely, it is necessary to configure ouf_fetch for the specific links that it should follow.

The `LinkTypes` option in the `[FetchRequest]` section of the ouf.ini file can be configured with a list of the link types within a title which ouf_fetch should follow or ignore.

The syntax of the LinkTypes option is a comma and/or white space delimited list of Tokens.

In this context, a Token is a two-character string with an optional minus sign.

A Token can be an explicit representation of a known type, such as "Tp," but can also contain a the wildcard character '*' as either the first, second or both characters.

For example, the Token "T*" would mean "all link types that begin with 'T'" (e.g. Ta, Tb, Tc…), and similarly, "*a" would mean all link types ending with 'a' (e.g. Ta, Aa, Ha etc).

By logical extension, "**" means match all link types.

A minus sign in front of a Token means "Ignore this Token," e.g. "-T*" means ignore all link types that begin with 'T', and "-**" would mean "ignore all links," which is actually the default behavior, anyway.

The list in the LinkTypes option read from left to right, and the first match or rejection of a link type is considered valid.

LinkTypes Examples

The examples below should clarify the configuration and handling of the LinkTypes option:

Example list: Ta, Tb

The above list will match link types Ta and Tb and nothing else.

Example list: -Ta, **

The above list will match everything except Ta. Not that this is completely different from **, -Ta which be the "left-to-right first" rule, would match everything, including Ta.

Example list: -**

The above list would reject all link types, which is the default behavior.

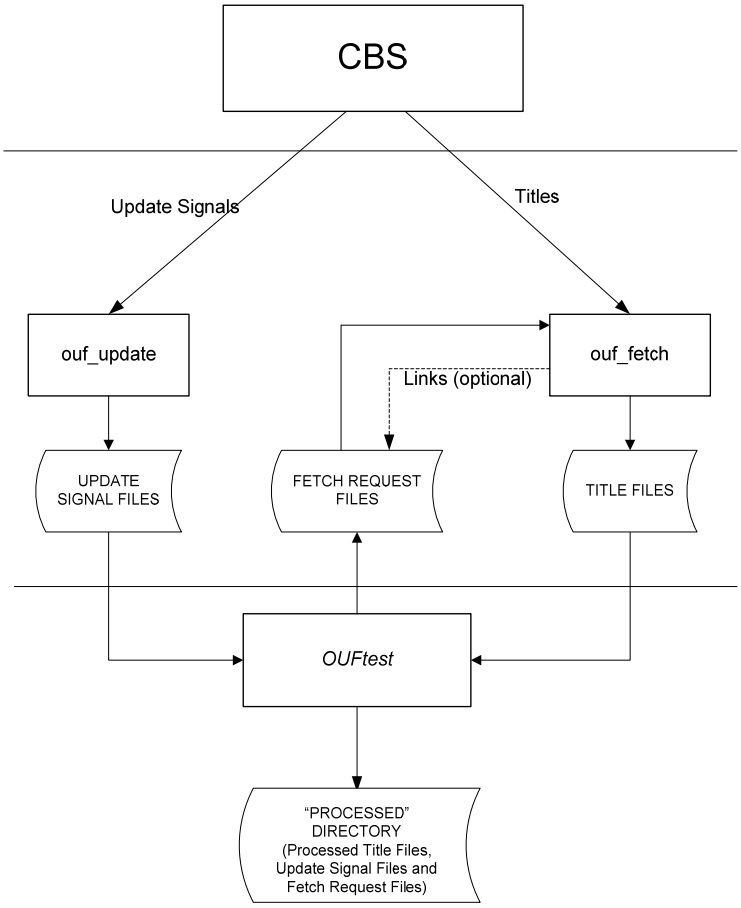
Appendix D: *The OUFTest Update Process Simulator*

Introduction to OUFTest

The OUFTest program was developed in order to test the interface "underneath" the OUF application(s) to a user Update Process.

OUFTest interprets Update Signal Files created by ouf_update, uses that information to create Fetch Request Files for ouf_fetch and finally reads the Title Files that ouf_fetch creates from the fetched titles.

The following diagram is essentially the same as the OUF process flow diagram in Chapter 2 of this document, with the difference that the Update Process has been replaced by OUFTest, and that the user's local database has been replaced by a "Processed" directory:



Configuring OUFTest

Introduction

The configuration of OUFTest is similar to that of OUF.

OUFTest reads a configuration file named `ouftest.ini` which has the same basic structure as ouf.ini.

A description of each section and the available options follows:

The `[Trace]` Section

The OUFTest `[Trace]` section contains a subset of the parameters available in the OUF configuration file:

Trace	Enables/disables tracing: *ON* enable tracing. *OFF* disable tracing.
Dir	The name of the directory where the trace file will be created
Screen	*Screen* Enables tracing to the display screen. Two values can be used: *OFF* disable screen output. *ON* enable screen output.

Example:

```
[Trace]
Dir=/home/ouf/trace
Screen = OFF
Trace = ON
```

Trace file names for OUFTest are generated using the same **`trace_DDMMYYYYHHMMSSCCC`** paradigm as used in OUF.

In this example tracing is enabled, and the trace file will be created in the directory `/home/ouftest/trace`. Tracing is set to file only. *See appendix E for an excerpt of an OUFTest trace fle..*

The [Directories] Section

Like OUF, OUFTest must be configured with a Fetch Request Directory, an Update Signal Directory and a Title Directory. These three directories should obviously configured to the same values as their counterparts in OUF for the interaction between OUF and OUFTest to work as intended.

An additional directory which is used only by OUFTest is the so-called "Processed" directory.

Since OUFTest simulates the behavior of a user Update Process, it must read Update Signal Files generated by ouf_update, create Fetch Request Files for ouf_fetch, and interpret the Title Files written by ouf_fetch.

As part of the simulation, OUFTest moves the Update Signal Files and Title Files that it has processed to its own Processed Directory, where the user can inspect the files later. OUFTest also creates copies in the Processed Directory of every Fetch Request File that it creates for ouf_fetch.

The following table shows the options in the `[Directories]` section of the configuration file:

Processed Dir	*Processed Dir* sets the Processed Directory where OUFTest stores copies of the fetch request files it generates and Update Signal Files created by ouf_update and the Title Files created by ouf_fetch.
Update Signal Dir	*Update Signal Dir* is the directory where OUFTest retrieves the Update Signal Files which have been generated by ouf_update.
Fetch Request Dir	The *Fetch Request Dir* is where OUFTest writes Fetch Request Files for processing by ouf_fetch.
Title Dir	The *Title Dir* is where OUFTest expects to find any Title Files written by ouf_fetch.

Example:

```
[Directories]
Processed Dir = /home/ouf/proc
Update Signal Dir = /home/ouf/updsig
```

```
Fetch Request Dir = /home/ouf/fetch
Title Dir = /home/ouf/data
```

OUFTest Procedures

The following describes the tasks performed by OUFTest:

OUFTest runs in a processing loop with following steps

Check For a Stop Signal.

OUFTest can be stopped by an external signal. If this signal has been received, then the program shuts down in a controlled fashion.

Check For Update Signal Files and Process One File, If Present.

The Update Signal Directory is checked for the presence of an Update Signal File. OUFTest processes the first file (which is not necessarily the oldest or newest file) in the directory in the following fashion:

a) it moves the Update Signal File to the Processed Directory.

b) it logs the timestamp, filename and contents of the Update Signal File.

c) it creates a Fetch Request File in the Processed Directory containing all of the PPNs (possibly with ILN) from the Update Signal File that had updates of the type *IN* or *MU*.

d) it copies the completed Fetch Request File to the Fetch Request Directory.

e) it logs the timestamp, filename and contents of the Fetch Request File.

Check For Title Files and Process One File, If Present

The Title Directory is checked for Title Files written by ouf_fetch. OUFTest processes the first file found, if any, in the following fashion:

a) it moves the Title File to the Processed Directory.

b) it logs the timestamp, filename and contents of the Title File The summary consists of the headers of the title records in the file.

c) it scans all global subrecords (dbslvl=BIB) of all title records for references to other records. These references are coded in $9 subfields. It stops if no references are found.

d) it creates a Fetch Request File in the Processed Directory with a maximum of 5 PPN references.

e) it copies the Fetch Request File to the Fetch Request Directory.

f) if logs the timestamp, filename and contents of the Fetch Request File.

Sleep For Five Seconds

If there are no Update Signal Files or Title Files to process, then the OUFTest process goes to sleep for five seconds before waking up and checking the Update Signal Directory and Title Directory again for files to process.

Starting OUFTest

OUFTest can in principle run continuously as a background process, so one way to start OUFTest is shown below:

```
> ouftest&
[1] 22654
```

If OUFTest is successfully started the ID of the OUFTest process is returned. This process ID is used later when signaling OUFTest to stop running (e.g. `kill 22654`). However, if an error occurred, then the error is written both to standard output and the error log file, and the OUFTest process will be stopped. In the following table all error messages that can occur when starting OUFTest are explained.

6.4.1.1.1.1 Fetch request directory not found	The fetch request directory was not found. Check the directory name contained in the `Fetch Request Dir` field of the `[Directories]` section in `ouftest.ini`.
Fetch request directory is write-protected	OUFTest uses the fetch request directory to store fetch request files while processing titles and update signals, so the directory must be made writable.
FILEMAP not present	The `FILEMAP` file was not found. This file contains settings for Pica applications and should be located in the subdirectory `pica/confdir` within the home directory.
Profile not opened	The profile file `ouftest.ini` was not

	opened, probably because the file does not exist in the `confdir` subdirectory.
Profile environment variable not set	The environment variable `DOTINIDIR` was not found in the `FILEMAP` file. This variable contains the name of the directory where the profile file (`ouftest.ini`) can be found and should be set to `$PICAMAINDIR/confdir`.
Processed directory not found	The processed directory was not found. Check the directory name contained in the `Processed Dir` field of the `[Directories]` section in `ouftest.ini`.
Processed directory is write-protected	OUFTest uses the processed directory to store data while processing title files and update signals, so the directory must be made writable.
Title directory not found	The title directory was not found. Check the directory name contained in the `Title Dir` field of the `[Directories]` section in `ouftest.ini`.
Title directory is write-protected	OUFTest uses the title directory to retrieve title files which are moved to the processed directory after processing, so the title directory must be made writable.
Trace file was not created	The trace file was not created probably because the directory in which the trace file should be written is not writable or does not exist. Check the directory name contained in the `Dir` field of the `[Trace]` section in `ouftest.ini` and also check the permission mode of this

	directory.
Trace file was not opened	The trace file was not created probably because the directory in which a previous trace file is stored is not writable or does not exist. Check the directory name contained in the `Dir` field of the `[Trace]` section in `ouftest.ini` and also check the permission mode of this directory.
Update signal directory is write-protected	OUFTest uses the update signal directory to retrieve update signals which are moved to the processed directory after processing, so the update signal directory must be made writable.
Update signal directory not found	The update signal directory was not found. Check the directory name contained in the `Update Signal Dir` field of the `[Directories]` section in `ouftest.ini`.

Stopping OUFTest

Once the process is started it runs until it is stopped by sending a stop signal. OUFTest should not be aborted by means of a break signal, e.g. by pressing <ctrl>+<c>. When OUFTest is aborted, the processing of titles and update signals will be terminated, leading to incomplete fetch request files in the processed directory.

In order to stop OUFTest in the proper way, the following command should be issued:

```
> kill 22654
> OUFTest was successfully terminated.

[1+] Done                 ouftest
>
```

OUFTest will display a message when terminated that indicates if any errors occurred during execution. If an error did occur, it will be displayed on the next line. In general,

errors may occur only when OUFTest is started, e.g. when a particular directory does not exist.

Appendix E: *An Example Of an ouf_update Trace File*

The following is an excerpt from a trace fie generated by ouf_update:

```
ouf_update: initial timestamp: 14-10-2004.
ouf_update: settings initialized.
ouf_update: checking for required directories...
dir /users/michael/pica/ouf/update_d doesn't exist, trying to create it.
directory "/users/michael/pica/ouf/update_d" created.
ouf_update: required directories checked.
[25-01-2005, 10:12:09] Connecting to 'develop.pica.nl' (port: 1140)...
[25-01-2005, 10:12:09] Connected.
[25-01-2005, 10:12:09] Logging on...
[25-01-2005, 10:12:10] Logged on.
[25-01-2005, 10:12:10] Selecting database 'BES1.SYS1@server.domain'...
[25-01-2005, 10:12:10] Database selected.
[25-01-2005, 10:12:10] ouf_update: update signal file 'updsig_20050125101210000'
created.
[25-01-2005, 10:13:31]
ouf_update: update signal file 'updsig_20050125101210000' closed.
- Total # records   : 450
- Oldest timestamp  : 14-10-2004
- Youngest timestamp: 17-01-2005 11:16:07.910000 [25-01-2005, 10:13:31] ouf_update:
update signal file 'updsig_20050125101331000' created.
[25-01-2005, 10:13:32]
ouf_update: update signal file 'updsig_20050125101331000' closed.
- Total # records   : 750
- Oldest timestamp  : 17-01-2005 11:16:07.910000
- Youngest timestamp: 18-01-2005 12:06:12.100000 [25-01-2005, 10:13:32] ouf_update:
update signal file 'updsig_20050125101332000' created.
ouf_update: sleeping 30 secs
ouf_update: sleeping 30 secs
[25-01-2005, 10:14:32]
ouf_update: update signal file 'updsig_20050125101332000' closed.
- Total # records   : 62
- Oldest timestamp  : 18-01-2005 12:06:12.100000
- Youngest timestamp: 24-01-2005 16:10:40.560000
ouf_update: sleeping 30 secs
ouf_update: sleeping 30 secs
ouf_update: sleeping 30 secs
ouf_update: sleeping 30 secs
ouf_update: sleeping 30 secs
ouf_update: sleeping 30 secs
ouf_update: sleeping 30 secs
ouf_update: sleeping 30 secs
ouf_update: sleeping 30 secs
ouf_update: sleeping 30 secs
```

Appendix F: *An Example Of an ouf_fetch Trace File*

The following trace file was generated by ouf_fetch:

```
ouf_fetch: gConf initialized.
ouf_fetch: checking for required directories...
dir /users/michael/pica/ouf/fetch_d doesn't exist, trying to create it.
directory "/users/michael/pica/ouf/fetch_d" created.
dir /users/michael/pica/ouf/title_d doesn't exist, trying to create it.
directory "/users/michael/pica/ouf/title_d" created.
ouf_fetch: required directories checked.
[25-01-2005, 10:14:33] Connecting to 'develop.pica.nl' (port: 1140)...
[25-01-2005, 10:14:33] Connected.
[25-01-2005, 10:14:33] Logging on...
[25-01-2005, 10:14:33] Logged on.
[25-01-2005, 10:14:33] Selecting database 'BES1.SYS1@server.domain'...
[25-01-2005, 10:14:33] Database selected.

[25-01-2005, 10:17:33]
ouf_fetch: fetch request file 'fetch_2005012510170701' opened.
ouf_fetch: title file '/users/michael/pica/ouf/tmp/T_0009_20050125101743000' created.
ouf_fetch: title file '/users/michael/pica/ouf/tmp/T_0015_20050125101743001' created.
ouf_fetch: title file '/users/michael/pica/ouf/tmp/T_0041_20050125101743002' created.
ouf_fetch: title file '/users/michael/pica/ouf/tmp/T_0003_20050125101743003' created.
ouf_fetch: title file '/users/michael/pica/ouf/tmp/T_0220_20050125101743004' created.
ouf_fetch: title file '/users/michael/pica/ouf/tmp/T_0013_20050125101744000' created.
ouf_fetch: title file '/users/michael/pica/ouf/tmp/T_0001_20050125101747000' created.
ouf_fetch: title file '/users/michael/pica/ouf/tmp/T_0035_20050125101747001' created.
ouf_fetch: title file '/users/michael/pica/ouf/tmp/T_0000_20050125101747002' created.
ouf_fetch: title file '/users/michael/pica/ouf/tmp/T_0002_20050125101747003' created.

[25-01-2005, 10:17:48]
ouf_fetch: fetch request file 'fetch_2005012510170701' closed.
- Total # lines  : 188
- Total # records: 188

[25-01-2005, 10:17:48]
ouf_fetch: fetch request file 'fetch_2005012510170702' opened.

[25-01-2005, 10:18:07]
ouf_fetch: fetch request file 'fetch_2005012510170702' closed.
- Total # lines  : 628
- Total # records: 628

[25-01-2005, 10:18:07]
ouf_fetch: fetch request file 'fetch_2005012510170703' opened.
ouf_fetch: title file '/users/michael/pica/ouf/tmp/T_0025_20050125101807000' created.

[25-01-2005, 10:18:07]
ouf_fetch: fetch request file 'fetch_2005012510170703' closed.
- Total # lines  : 7
- Total # records: 7
```

Appendix G: *An Example of an OUFTest Trace File*

```
Settings initialized.
Checking for required directories...
Required directories checked.
Checking for update signal files...
Start processing update signals...
Initializing processing of update signals...
[25-01-2005, 10:17:07] Update signal file 'updsig_20050125101210000' opened.
Processing of update signals initialized.
Contents of file 'updsig_20050125101210000':
015 1999       15-10-2004 11:51:27.570.000 [MU] 810531119 0 00 Aa
015 1999       15-10-2004 11:51:56.280.000 [MU] 810531119 0 00 Aa
015 1999       15-10-2004 11:52:17.290.000 [MU] 810531119 0 00 Aa
015 1999       15-10-2004 11:53:30.250.000 [MU] 810531119 0 00 Aa
[…]
015 1999       17-01-2005 11:03:30.880.000 [MU] 242685722 2 02    394455274
015 1999       17-01-2005 11:03:30.890.000 [IN] 242685722 2 03    394656504
015 1999/0001  17-01-2005 11:16:07.820.000 [MU] 115622241 0 00 Aa
015 1999/0001  17-01-2005 11:16:07.910.000 [IN] 115622241 2 01    394656512
---- end of file ----
Creating PPN list...
PPN list created.
Sorting PPN list...
PPN list sorted.
Start writing fetch requests...
[25-01-2005, 10:17:07] Fetch request file 'fetch_2005012510170701' created.
[25-01-2005, 10:17:07] Fetch request file 'fetch_2005012510170701' closed.
Fetch requests written.
Freeing memory used by PPN list...
Memory freed.
[…]
Checking for title files...
Start processing titles...
Initializing processing of titles...
[25-01-2005, 10:18:12] Title file 'T_0009_20050125101743000' opened.
Processing of titles initialized.
Summary of contents:
PPN:039901947 009 P -> OK
PPN:040032574 009 P -> OK
[…]
PPN:19673004X 009 P -> OK
PPN:196730058 009 P -> OK
PPN:196730074 009 P -> OK
PPN:196730082 009 P -> OK
PPN:196730104 009 P -> OK
---- end of summary ----
Start scanning title file for references...
Title file scanned for references.
[25-01-2005, 10:18:13] Title file 'T_0009_20050125101743000' closed.
Issuing system command: mv /users/michael/pica/ouf/title_d/T_0009_20050125101743000
/users/michael/pica/ouf/processed_d
[25-01-2005, 10:18:13] Title file 'T_0009_20050125101743000' moved.
Finished processing titles.
Sleeping for 5 seconds...
```

Appendix H: *Building and Configuring the OUF Package*

Environment for OUF

In order to compile and run OUF, a special UNIX user account is needed.

This account should contain the following lines in its `.profile`:

```
PICAHOME=$HOME export PICAHOME
APPLMAINDIR=$PICAHOME export APPLMAINDIR
FILEMAP=$APPLMAINDIR/confdir/FILEMAP export FILEMAP
```

- The environment variable `PICAHOME` is used during `make`.
- The environment variable `FILEMAP` is used by OUF for initialization. It contains lines of the structure `NAME=value`, which may look like shell variable assignments, but are in fact interpreted by OUF on startup. This means that only simple assignments are possible.
- The environment-variable `APPLMAINDIR` is used inside the `FILEMAP` only.

Before compiling, there should be an executable shell script named `ccom` in the `$PATH`. This script should contain something along the following lines (the compilation flags depend on the compiler you use):

```
#! /usr/bin/ksh

INCLUDES=

# For Solaris you will need –lsocket and –lnsl in $LIBRARIES
LIBRARIES="-lpica"
LIBRARYDIR_ARGS="-L$PICAHOME/lib"

COMPILE_FLAGS="-D_POSIX_SOURCE"
DEB_FLAGS="-g"

for i in "$@" ; do
        case "$i" in
        -I*) INCLUDES="$INCLUDES $i" ;;
        -c) LIBRARIES= ;; # Avoid complaints about -l
        esac
done

INCLUDES="$INCLUDES -I$PICAHOME/include"

# The compiler can also be gcc
cc $COMPILE_FLAGS $DEB_FLAGS $LIBRARYDIR_ARGS $INCLUDES "$@" $LIBRARIES
```

Installing OUF

The distribution package is delivered in a compressed tar file which can be unpacked by the following command:

```
zcat ouf.tar.Z | tar xf -
```

This will create a subdirectory named `ouf`. This directory will contain a script called `install` which should be executed by the command `ouf/install`.

This script will first ask whether it should delete .DEL files. Since this is a new version, pressing `ENTER` will be sufficient.

After this, all of the directories needed will be created and the source tree will be copied to `$HOME/src`. After that, the script will indicate which `.example` files need to be changed to configure the package for the given system.

The command `make all` in directory `$HOME/src` will build the complete system, i.e. compile the C source files into the programs `ouf_fetch` and `ouf_update`.

Configuring OUF

Before `ouf_fetch` and `ouf_update` are run, they must be configured. This means that all .example files should be copied to files without the .example extension. This can be done using the following Bourne shell script in both the directories `$HOME/bin` and `$HOME/confdir`:

```
for FIL in `find . -name '*.example' –print`
do
   cp $FIL `echo $FIL | sed 's/.example$//'`
done
```

After that, all of the new files must be edited and adapted for the local configuration.

Appendix H: *The OUF / OUFTest Source File Distribution*

The following Makefile and C files are used to build OUF and OUFTest:

```
Makefile          Makefile for OUF
ouf_cbs.c         OUF CBS library routines
ouf_fetch.c       OUF fetch routines
ouf_pf.c          Configuration file routines
ouf_pf.h          Configuration file header
ouftest.c         OUFTest source code
ouftest.h         OUFTest header file
ouftest_main.c    OUFTest main function
ouftest_parse.c   OUFTest parser source
ouftest_parse.h   OUFTest parser header
ouf_trace.c       Trace file routines
ouf_trace.h       Trace file header file
ouf_upd.c         OUF update routines
ouf.h             Common OUF header file
ouf_util.c        OUF shared "utility" functions
```