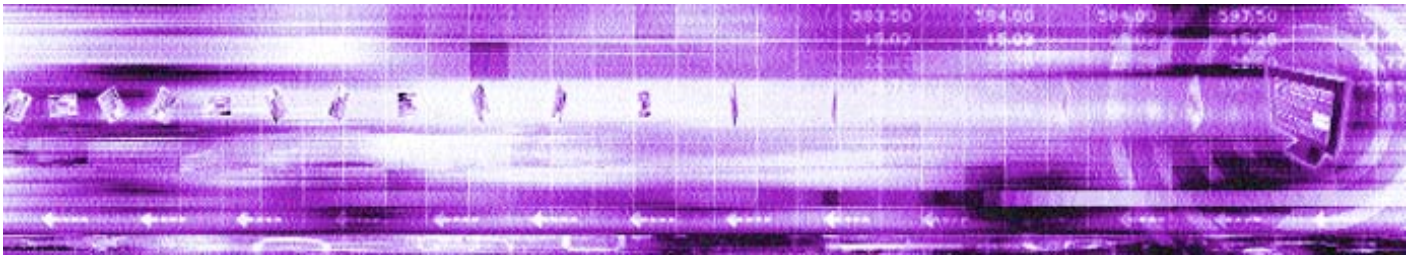

CAPLIN XML AUTH 4.4

Administration Guide

April 2007

1 Preface

1.1 What this document contains

This document describes the XML Auth Module and how it enables programmers and system administrators to use XML to create their own permissioning structures and control entitlement to objects held on Caplin Liberator.

For details on how to create your own Auth Modules, refer to the companion document SDK html.

1.2 Who should read this document

This document is intended for programmers and system administrators who use XML to create permissioning structures and control the entitlement of objects held on Caplin Liberator.

1.3 Typographical conventions

This document uses the following typographical conventions to identify particular elements within the text.

Type	Use
Arial Bold	Function names and methods. Other sections and chapters within this document.
Arial Italic	Parameter names and other variables.
Times Italic	File names, folders and directories.
`Courier`	Program output and code examples.
❖	Information bullet point.
■	Instruction.

1.4 Acronyms and glossary

Auth Module	An application that performs authentication and authorisation functions.
Authentication	Permitting a particular user to login to Caplin Liberator in order to access streaming data. Also known as entitlement.
Authorisation	Permitting particular data to be viewed.
Caplin Liberator	A suite of software applications and components for publishing real-time information using RTTP protocol over IP networks. Caplin Liberator collects real-time data from one or more sources and redistributes it to suitably permissioned RTTP subscribers.
DACS	Reuters's system for administering market data and permissioning trading floor, enterprise and Internet users to data, transactions, and other services.
DTD	Document Type Definition, a specification that accompanies an XML document and identifies what the tags are that separate paragraphs and how each is to be processed.
Field	A data element of an object, identified by a field name or field number and with a data value of a string.
Object	There are several types of RTTP object: Directory, Page, Record, News headline, News story and Chat object. Each type is identified by a three digit number.
Parameter	A data element of an object, identified by a field name or field number and with a data value of a string. Same as "field".
Parser	A program that receives input in the form of program instructions, markup tags or some other interface and breaks them up into parts (for example, objects, methods and their attributes) that can then be managed by other programs.

RTTP	RTTP (Real Time Text Protocol) is a web protocol developed by Caplin Systems Ltd that implements advanced real-time streaming for almost all types of textual information, including logical records, news and free-format pages.
Symbols	The letters used to uniquely identify a financial instrument (e.g. the symbol for Microsoft's common stock on the Nasdaq market is MSFT). Many symbol naming conventions (symbologies) exist, but each is applied consistently in each market, and one symbology is used across all North American equity markets.
UDP	UDP (User Datagram Protocol) is a communications protocol that offers a limited amount of service when messages are exchanged between computers in a network that uses the Internet Protocol.
XML	XML (Extensible Markup Language) contains markup symbols to describe the contents of a page or file. An XML file can be processed purely as data by a program or it can be stored with similar data on another computer or displayed. XML is "extensible" because, unlike HTML, the markup symbols are unlimited and self-defining.
XML tags	The sequence of characters or other symbols that you insert at certain places in a file to indicate how the file should look when it is printed or displayed or to describe the document's logical structure. Tags that have other tags within them are called parent tags; those within are called child tags.

1.5 Feedback

Customer feedback can only improve the quality of Caplin product documentation, and we would welcome any comments, criticisms or suggestions you may have regarding this document.

Please email your thoughts to documentation@caplin.com.

2 Getting Started

2.1 What is the XML Auth Module?

Caplin Liberator supports a modular system for handling authentication of users and entitlement to objects. This allows users to be authenticated, objects to have permissions loaded, read and write permissions for a user to be checked and object name mappings to be performed.

An Auth Module provides a means of authenticating users and authorising access to objects on a modular basis.

The XML Auth Module was initially developed to integrate Reuters DACS content-based permissioning databases with Liberator, but it is not tied to handling data from DACS.

Note: *DACS (Data Access Control System) is a feature of the Reuters Market Data System (RMDS) that provides client administrators with the ability to control user access to information vendors, products, exchanges and specialist data services, as well as internally published data. DACS controls realtime data access by users (i.e. any entity with an operating system login). Users can be permitted to retrieve data from any workstation in the market data system subject to a maximum number of simultaneous logins.*

The Caplin Platform architecture

Figure 2-1 below shows a detailed illustration of Caplin's platform architecture, including all available products and the protocols they use to communicate.

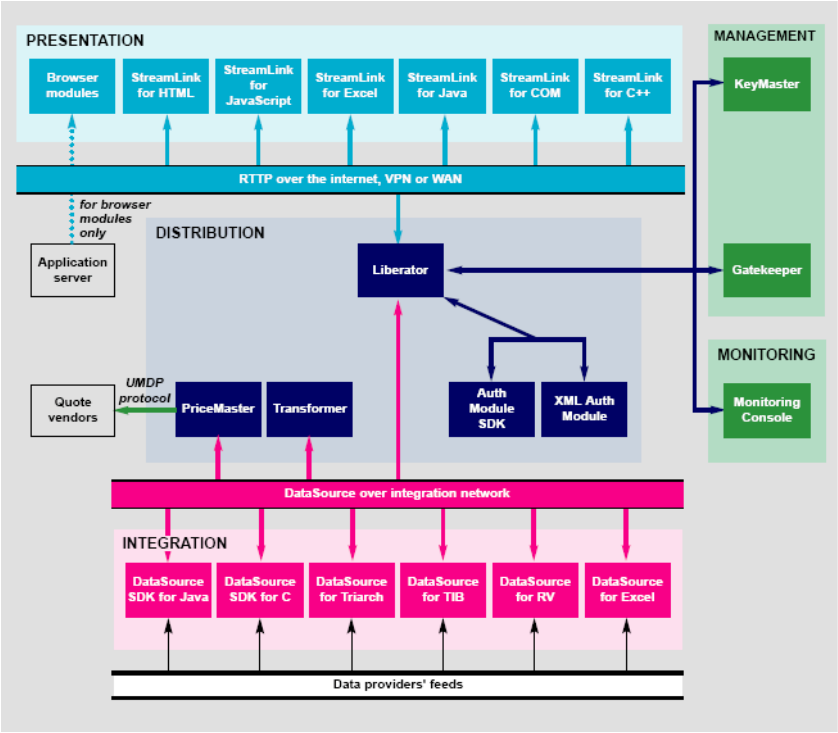

Figure 2-1: Caplin's platform architecture

Figure 2-2 below shows a simplified Caplin platform diagram and highlights the XML Auth Module, the Auth Module SDK and the Liberator.

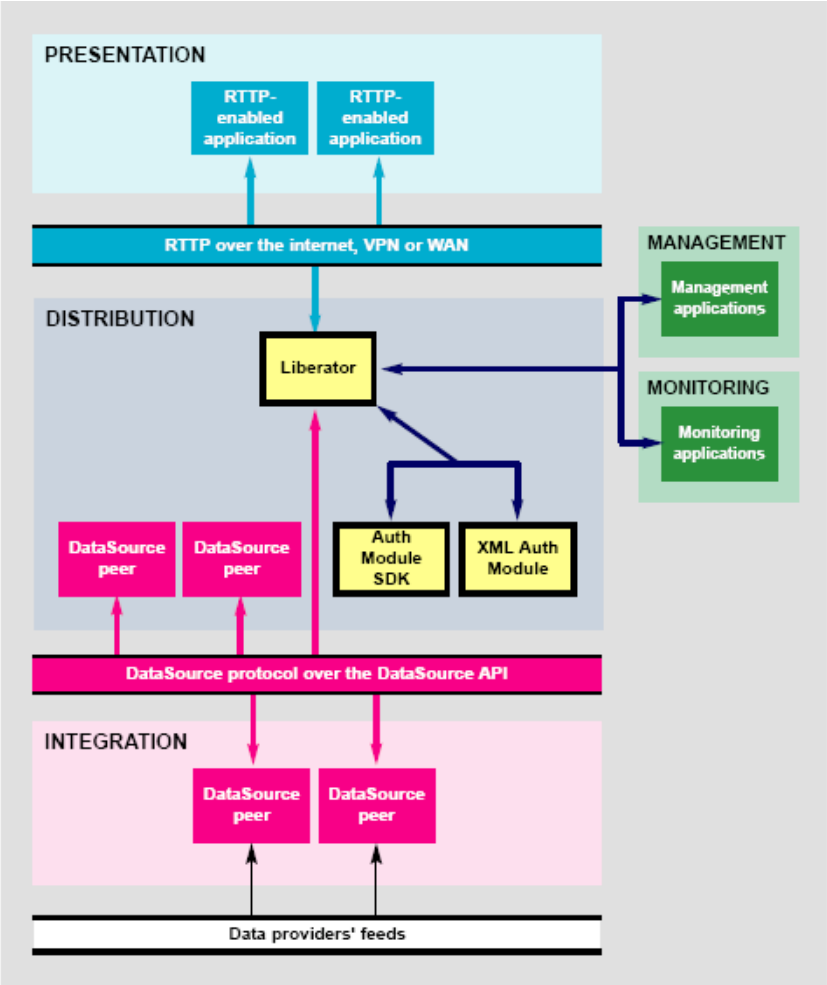

Figure 2-2: Auth Modules' place in Caplin platform architecture

In order to use the XML Auth module you need two components:

- **XML Auth Module** - The means to authenticate users, load their permissions for objects, check their read and write permissions and perform object name mappings.

- **XML Auth Module parsing library** - XML Auth Module is built upon the Expat XML parsing library, a stream-oriented parser in which an application registers handlers for things the parser might find in the XML document (like start tags). For more information, refer to http://expat.sourceforge.net.

Both of these components are installed during the Liberator installation, so no specific actions or requirements are necessary to start using XML Auth Module.

3 Configuring XML Auth Module

XML Auth Module is configured by editing the following entries in the file xmlauth.conf within the Caplin Liberator installation directory. An example of xmlauth.conf is included as Appendix E starting on page 41.

3.1 Configuring the user file

An example user file is included on page 24. The XML tags that can be used in a user file are described on page 21. The associated user .dtd file is included as Appendix A starting on page 32.

user-file

Name and location of the .xml file containing details of users' definitions. The filename should include the full path, and we recommend that the file is put in the etc/ directory, as this is where certain Caplin products will look for it.

Default value: [no default]

user-file supports the following abbreviations:

%r	Liberator configuration parameter application-root (the root directory of the application installation)
%a	Liberator configuration parameter application-name (a name to distinguish the Liberator from other applications)
%h	The host machine name.

For more information on Liberator configuration refer to the **Liberator Administrator's Guide**.

default-user

Specifies the name of a default user. XML Auth Module default behaviour is to log in a client with the default user's permissions if an invalid login is given. This can be overridden using the default-perm configuration option.

Default value: NULL

default-perm

Boolean to indicate that default-user's permissions are added to every user's permissions. This means that if a particular user has no permission to view a certain object but the default user does, the user will be granted permission.

Setting default-perm overrides the default behaviour of using default-user's permissions if the user gives an invalid login.

Default value: FALSE

3.2 Configuring the entity file

An example entity file is included on page 27. The XML tags that can be used in an entity file are described on page 27. The associated entity .dtd file is included as Appendix B starting on page 34.

entity-file

Name and location of the .xml file containing details of entities' definitions. Entities are individual fields for which permissions can be granted. An entity can also be a specific value for an individual field. The filename should include the full path, and we recommend that the file is put in the Caplin Liberator's etc/ directory, as this is where certain Caplin products will look for it.

Default value: [no default]

entity-file supports the following abbreviations:

%r	Liberator configuration parameter application-root (the root directory of the application installation)
%a	Liberator configuration parameter application-name (a name to distinguish the Liberator from other applications)
%h	The host machine name.

For more information on Liberator configuration refer to the Liberator Administration Guide.

numeric-entity

Indicates that entities are wholly numeric.

Entities control the content-based permissioning aspect of the module, and can be either wholly text, wholly numeric, or mixed. If they are wholly numeric this option should be enabled, as performance can be improved by treating entities as numbers and not strings.

Default value: false

3.3 Configuring additional files

XML Auth Module can permission users by parsing additional XML files containing user and entity information. This information must use the same tags used in user and entity files (see **Creating Permissions** on page 20).

add-file

Name and location of the .xml file containing additional user and service definitions. The filename should include the full path and should be located in the etc/ directory. You can use an add-file entry for each relevant file.

Default value: [no default]

add-file supports the following abbreviations:

%r	Liberator configuration parameter application-root (the root directory of the application installation)
%a	Liberator configuration parameter application-name (a name to distinguish the Liberator from other applications)
%h	The host machine name.

For more information on Liberator configuration refer to the Liberator Administrator's Guide.

3.4 Reloading user and entity files

reload-time

Time of day to reload any changed .xml files in form HH:MM:SS

Default value: NULL

3.5 Adjusting XML Auth Module's performance

hashtable-size

The size of the hash table used to store the information. Giving this parameter a larger value can improve performance in some circumstances.

Default value: 16384

3.6 Ejecting users

eject-after-read

If set then all users' sessions are invalidated after reading the XML permission files by sending the UDP command xmlauth-ejectall. If the user is no longer permissioned they are logged out of the system.

See page 30 for more information on invalidating sessions and the xmlauth-ejectall command.

Default value: FALSE

auth-application-eject

Boolean parameter. Enables user ejection based on the ID of the application they are using, so that a user currently at their application session limit will be ejected from the first session when logging in again, but all remaining sessions will continue unaffected.

Default value: FALSE

auth-machine-eject

Boolean parameter. Enables machine-based user ejection, which means that a user's other sessions are ejected when a user logs on from a different machine.

Default value: FALSE

3.7 Auditing users

To assist with auditing users the XML Auth Module can write out an optional audit trail to a log file.

Entries in the audit log take the following in tab-separated format:

- Symbol
- Transaction type (either REQuest or DIScard)
- Date and time
- Logged-in username
- Username used for permissioning
- User location
- Application ID
- Session ID
- Host IP address
- Type of permissioning (either Subject-based, simple Content-based or Advanced content-based)
- Permissioning fieldname
- Value in permissioning field
- Entity name and entity ID if using content-based permissioning

Example of the audit log:

```
/I/VOD.LREQ ->  1038328872 ->   admin ->        admin ->        Loc1 ->
CLIENT/CLEAR -> 0grOYM ->       127.0.0.1 ->    S ->            PROD_PERM ->
5625 ->         - -> - ->       - ->            - ->
/I/VOD.L ->     DSC ->          1038328880 ->   admin ->        admin ->
Loc1àCLIENT/CLEAR -> 0grOYM ->  127.0.0.1 ->    PROD_PERM ->    5625 ->
- ->            - ->            - ->            - ->
```

If the transaction was a request and access was granted using subject-based permissioning then the value in the field identified by (subject-fieldname) (see page 17) is recorded in the log file.

audit-enable

Boolean parameter. Enables user auditing.

Default value: FALSE

audit-logfile

Name of the log file to record user audit information written to the audit log.

Default value: audit-%a.log

audit-logfile supports the following abbreviations:

%r	Liberator configuration parameter application-root (the root directory of the application installation)
%a	Liberator configuration parameter application-name (a name to distinguish the Liberator from other applications)
%h	The host machine name.

For more information on Liberator configuration refer to the **Liberator Administration Guide**.

subject-fieldname

Name of the field to extract audit value from. This field should be defined in fields.conf.

Default value: PROD_PERM

3.8 Permissioning users for news stories

By default users are allowed to read all news headlines. The following parameter in xmlauth.conf enables users to be permissioned for news headlines on a content basis.

auth-check-update Boolean parameter. Enables news permissioning.

Default value: FALSE

3.9 Using a password file

Users' passwords and number of logins permitted for each user can optionally be placed into an additional file named *passwd.xml*. *passwd.xml* can be updated separately to the user file and then the udp command xmlauth-reload issued to reload the user information.

An example password file is included on page 29. The XML tags that can be used in a password file are described on page 28. An example password .dtd file is included as **Appendix C: et_passwd.dtd** on page 35.

passwd-file Location of the password file *passwd.xml*. This file overrides *users.xml*—if a user is removed from passwd.xml then their settings in *user.xml* are used instead.

If this file cannot be found then passwords and licences will be derived from the information given in *users.xml*.

Default value: passwd.xml

passwd-file supports the following abbreviations:

%r	Liberator configuration parameter application-root (the root directory of the application installation)
%a	Liberator configuration parameter application-name (a name to distinguish the Liberator from other applications)
%h	The host machine name.

For more information on Liberator configuration refer to the **Liberator Administration Guide**.

3.10 XML Auth Module logging

debug-level

Determines the errors and events that are reported to the log files when XML Auth Module is operating. Log files are written to the same directory as Caplin Liberator log files—see the **Liberator Administration guide** for further information. Acceptable values of debug-level are shown in Table 3-1 below.

If the UDP message interface (which is configured in the Liberator configuration file) is enabled then the logging level can be changed whilst XML Auth Module is running. For details on how to achieve this, see xmlauth-debug on page 30.

Note: *A list of all error messages and their associated debug level can be found as* ***Appendix D: Debug levels and messages*** *on page 36.*

Default value: info

Value	Description
DEBUG	Reports all errors and events.
INFO	Reports events and information regarding normal operation and all errors included in the WARN, NOTIFY, ERROR and CRIT debug levels.
WARN	Reports minor errors and all errors included in the NOTIFY, ERROR and CRIT debug levels.
NOTIFY	Report errors regarding data corruptions and all errors included in the ERROR and CRIT debug levels.
ERROR	Reports serious errors regarding network connections and all errors included in the CRIT debug level.
CRIT	Reports critical errors that prevent XML Auth Module running.

Table 3-1: Debug levels

4 Creating Permissions

4.1 Permissioning methods

There are five ways that a user can be granted permission to view an object.

- **Subject-based permissioning**. A user is permitted to view an object if it is for a particular symbol (identified using the subject attribute in a <PERM> tag—see page 22).
- **Content-based permissioning**. A user is permitted to view an object if a specified field in the object has a particular value (specified by an entity type of INTERNAL containing particular <VALUE> tags—see page 25).
- **Content-based permissioning with cross matching**. A user is permitted to view an object if the specified field exists as both a FEELIABLE entity, and as a PRODUCT entity - see page 25).
- **Composite entities**. If a symbol's permission code falls within a COMPOSITE type entity then the user must be permissioned for all the sub-entities within that entity.
- **Product-only permissioning**. If a user is permissioned for a product and the permissioning field value is not found in any FEELIABLE entity then the user is allowed access.

4.2 XML Auth Module files

XML Auth Module uses three files to store and structure entitlement information.

These are:

- a file which contains details of users (identified by the user-file configuration option—see page 12)
- a file which contains details of entities (identified by the entities-file configuration option—see page 13). Entities are individual fields for which permissions can be granted. An entity can also be a specific value for an individual field.
- a file which contains details of users' passwords and number of logins permitted for each user (identified by the passwd-file configuration option—see page 18).

Each file has a corresponding document type definition (.dtd) file. The contents of these files are included as appendices, as shown in Table 4-1

File	DTD filename	Location
User file	et_users.dtd	Appendix A, page 32
Entity file	et_entities.dtd	Appendix B, page 34
Password file	et_passwd.dtd	Appendix C, page 35

Table 4-1: Location of DTD files

XML tags that can be used in these files are listed below.

4.3 User file tags

The following XML tags can be used within the user file to grant permissions.

<ET_USERS> Root tag for user file.

<USER> Defines a user, including whether this user should be granted access using Caplin's Keymaster.

Keymaster works by having a public key specified by an add-sigkey entry in rttpd.conf (see **Caplin Liberator Administration Guide** for more information). *add-sigkey* includes a user parameter, which must be the same as the siguser parameter below to authenticate a user.

You can have several users pointing to the same signature checking key by having each user's siguser as the same as the user parameter in an add-sigkey entry in *rttpd.conf*.

The following attributes can be used within a <USER> tag.

Name	Default	Description
name	[no default]	User name.
pass	[no default]	Required password.
logons	[no default]	Number of times this user can login to Caplin Liberator concurrently (i.e. the licence count).

Name	Default	Description
sigcheck	FALSE	If set to TRUE ignores pass and uses the signature checker to authenticate the user.
sigkey -id	name	If sigcheck is set to TRUE, identifies the user for the purpose of checking signatures.
expiry	[no default]	Time when this user's access to Liberator is cancelled. May be represented in one of the following ways: number of seconds after midnight January 1 1970 YYYY-MM-DD HH:MM:SS YY-MM-DD, with the year being implicitlly 20nn.
admin	FALSE	Boolean indicating that this user can view monitoring and system management information. This permissioning must be enabled using the auth-monitoring configuration option for this to take effect. This functionality is available from version 4.0 onwards.
location or site	[no default]	Location of user.

Example:

```
<USER name="demouser" pass="AO12345z" logons="1" expiry="05-12-12"
site="London">
```

<PERM> Defines a permission for the user. Must be used as a child tag within a <USER> tag.

The following attributes can be used within a <PERM> tag:

Name	Default	Description
subject	[no default]	Semi-colon-separated list of symbols for which permission is to be granted or revoked. Use asterisk (*) as a wildcard for any number of characters, question mark (?) as a wildcard for an individual character. Setting subject="*" will match all symbols
action	[no default]	Action to be taken when update for subject arrives. Acceptable values are: "grant" "readwrite" "rw" — User has read write access to the permissioned data "read" "r" — User has read access only to the permissioned data "write" "w" — User has write access only to the permissioned data "revoke" — No read or write access

Example:

```
<PERM subject="/I/*" action="grant">
```

<ENTITY>

Defines an entity from the entities file when used as a child tag within a <PERM> tag.

The specific entity name should be included between the opening and closing tags.

Example for the entity Test:

```
<ENTITY>TEST</ENTITY>
```

<MAP>

Defines an object mapping. Object mapping changes the internal name of an object when a user requests it, allowing a username to be included in the object name in order for each user to get a unique object. For example if userX requests /HN/NEWSSTORY/1234, the object could be mapped to /HN/NEWSTORY/userX/1234.

The following attributes can be used within a <MAP> tag:

Name	Default	Description
from	[no default]	Name of object to be changed. %u can be used as the username.
to	[no default]	New name for object. %u can be used as the username.

Example:

```
<MAP from="/HN/NEWSSTORY/"to="/HN/NEWSSTORY/%u/" />
<MAP from="/MYCHANNELS/%1"to="/CHANNELS/%u/%1" />
<MAP from="/ABC/%1/%2"to="/DEF/%2/%1" />
```

where %u is the username and %1 and %2 are strings to be matched in the pattern.

Example user file

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ET_USERS SYSTEM "et_users.dtd">

<ET_USERS>

<USER name="admin" pass="admin" logons="2">
      <PERM subject="*" action="grant" />
</USER>

<USER name="demouser" pass="demopass" logons="100">
      <PERM subject="/SYSTEM" action="revoke" />
      <PERM subject="*" action="grant" />
</USER>

</ET_USERS>
```

4.4 Entity file tags

The following XML tags can be used within the entity file to grant permissions for specific fields and field values.

<ET_ENTITIES> Root tag for entity file.

<ENTITY> Defines an entity. An entity is an individual field for which a permission is granted. You can enable permission to be granted for particular values of this field using <VALUE> tags as child tags within an <ENTITY> tag.

The following attributes can be used within an <ENTITY> tag.

Name	Default	Description
name	[no default]	Name of the entity.
field	[no default]	Name of the field to be permissioned.
type	INTERNAL	Entity type. Acceptable values are shown in Table 4-2

Type	Description
INTERNAL (or no type attribute)	Permission will be granted if an update for field arrives with a value identified in any of the following <VALUE> tags.
PRODUCT	Permission will be granted if an update for field arrives with a value identified in any of the following <VALUE> tags only if another entity of FEELIABLE type exists for this symbol.
FEELIABLE	Permission will be granted if an update for field arrives with a value identified in any of the following <VALUE> tags only if another entity of PRODUCT type exists for this symbol.
SPECIAL	Treated as PRODUCT.
COMPOSITE	User must be permissioned for all the sub-entities within that entity

Table 4-2: Entity types

<VALUE> Defines a value for the entity's field. Must be used as a child tag within an <ENTITY> tag.

The specific value should be included between the opening and closing tags.

Example for a value of 2849:

```
<VALUE>2849</VALUE>
```

Example entity file

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ET_ENTITIES SYSTEM "et_entities.dtd">

<ET_ENTITIES>

<ENTITY name="INTERNAL_DATA" field="INTERN_PG">
      <VALUE>1</VALUE>
      <VALUE>2</VALUE>
      <VALUE>3</VALUE>
</ENTITY>

<ENTITY name="DELAYED_DATA" field="INTERN_PG">
      <VALUE>4</VALUE>
      <VALUE>5</VALUE>
      <VALUE>6</VALUE>
</ENTITY>

<ENTITY name="NYSE" field="PROD_CATG">
      <VALUE>2001</VALUE>
      <VALUE>2002</VALUE>
      <VALUE>2003</VALUE>
</ENTITY>
<ENTITY name="CBOT" field="PROD_CATG">
      <VALUE>2004</VALUE>
      <VALUE>2005</VALUE>
      <VALUE>2006</VALUE>
</ENTITY>
<ENTITY name="NASDAQ" field="PROD_CATG">
      <VALUE>2007</VALUE>
      <VALUE>2008</VALUE>
      <VALUE>2009</VALUE>
</ENTITY>
<ENTITY name="LSE" field="PROD_CATG">
      <VALUE>2010</VALUE>
      <VALUE>2011</VALUE>
      <VALUE>2012</VALUE>
</ENTITY>
```

```
<ENTITY name="LIFFE" field="PROD_CATG">
      <VALUE>2013</VALUE>
      <VALUE>2014</VALUE>
      <VALUE>2015</VALUE>
</ENTITY>
<ENTITY name="MONEY_2000" field="PROD_CATG">
      <VALUE>2016</VALUE>
      <VALUE>2017</VALUE>
      <VALUE>2018</VALUE>
</ENTITY>

</ET_ENTITIES>
```

4.5 Password file tags

The following XML tags can be used within the password file to identify users and their passwords.

<ET_PASSWD> Root tag for user file.

<USER> Defines a user.

The following attributes can be used within a <USER> tag.

Name	Default	Description
name	[no default]	User name.
logons	[no default]	Number of times this user can login to Caplin Liberator concurrently (i.e. the licence count).
pass	[no default]	Required password.

Example:

```
<USER name="demouser" pass="AO12345z" logons="1" expiry="05-12-12"
admin site="London">
```

Example password file

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ET_PASSWD SYSTEM "et_passwd.dtd">
<ET_PASSWD>
<USER name="admin" pass="admin" logons="2" />
<USER name="demouser" pass="demopass" logons="100" />
</ET_PASSWD>
```

5 Using UDP commands

XML Auth Module includes a UDP command interface that enables you to send UDP commands and messages regarding debugging and the writing of lists of watched objects to files.

The UDP interface is configured by editing the udp-port and udp-interface entries in the *rttpd.conf* file within the Caplin Liberator installation directory.

5.1 UDP commands

UDP instructions can be sent over XML Auth Module's UDP interface the following command.

udpsend

Sends a UDP message.

Syntax: udpsend [-s <hostname] [-p <port>] <message>

Parameters:

Name	Type	Default	Description
-s	string	127.0.0.1	IP address of machine to send message to.

Name	Type	Default	Description
-p	integer	10001	Port that host machine listens on for UDP messages.
message	string	[no default]	Message to send. Possible values are: **xmlauth-debug <level>** Dynamically changes the level of error and event reporting. **xmlauth-reload** Reloads all XML files after they have been changed (see page 33). **xmlauth-ejectall** Invalidates all users sessions. This is optional after the user and entity XML files have been changed and reread and is activated by setting the eject-after-read configuration option (see page 15). Invalidating a user session involves XML Auth Module rerequesting details of what the user is looking at from Caplin Liberator and granting permissions accordingly. This enables the Auth Module to adjust a user's permissions in the middle of a session, and happens the next time an update for a symbol occurs. ❖ If the user is no longer entitled to see a packet then the client gets a DELETED message. ❖ If the user no longer exists then they are disconnected from the system. If xmlauth-ejectall is not used then permissions are checked only when a user requests a new symbol. If the user no longer exists then they will be disconnected, otherwise they will get the normal available/not available messages.

Example:

```
udpsend -s 127.0.0.1 -p 10001 xmlauth-debug NOTIFY
```

6 Appendix A: et_users.dtd

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- DTD for user-based entitlements -->

<!ELEMENT ET_USERS (USER*)>

<!-- one entry for each user -->
<!-- logons is an optional field containing the number of
concurrent
     logons that this user is allowed to have.
       default will be to allow as many concurrent logons as
possible. -->
<!ELEMENT USER (MAP*,GROUPS*, PERM*)>
<!ATTLIST USER
      name        CDATA #REQUIRED
      logons      CDATA #IMPLIED
      pass        CDATA #IMPLIED
      sigkey -id  CDATA #IMPLIED
      sigcheck    CDATA #IMPLIED
      expiry      CDATA #IMPLIED
      admin       CDATA #IMPLIED
>

<!-- Request name mapping -->
<!ELEMENT MAP EMPTY>
<!ATTLIST MAP
    from          CDATA #REQUIRED
    to            CDATA #REQUIRED
>

<!-- the list of groups of perms to include -->
<!ELEMENT GROUPS (GROUP*)>
```

```
<!-- the perm group name to include -->
<!-- the action can be grant or revoke.
            this can be used to provide +ve or -ve permissioning --
>
<!ELEMENT GROUP (#PCDATA)>
<!ATTLIST GROUP
      action CDATA #REQUIRED

<!-- a user permissioning list.
      Many of these can be specified -->
<!-- the subject attribute is optional -
      specifies the valid subjects to apply this permissioning to.
      if no entities are listed, then can be used to specify
      pure subject-based entitlements.
      If no subject is specified, then assume that this user can
subscribe
      to all subjects and check this permissioning against it. -->
<!-- the action can be grant or revoke.
      this can be used to provide +ve or -ve permissioning -->
<!ELEMENT PERM (ENTITY*, RANGES*)>
<!ATTLIST PERM
      subject CDATA #IMPLIED
      action CDATA #REQUIRED
>

<!-- a single permissioning entity - its name -->
<!ELEMENT ENTITY (#PCDATA)>

<!-- a range list - e.g. for pages 100-200
        the subject_part is the part to apply the range to -->
<!ELEMENT RANGES (RANGE+)>
<!ATTLIST RANGES
      subject_part CDATA #REQUIRED>
<!-- a range  e.g. 100-200-->
<!ELEMENT RANGE (#PCDATA)>
```

7 Appendix B: et_entities.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- DTD for permissionable entities -->
<!-- last modified 8th April 2001 -->
<!-- 8/4/2002 - added optional type attribute -->
<!-- one entry for each entity -->
<!-- the field is the field to check the values against - e.g.
PROD_CATG -->
<!ELEMENT ET_ENTITIES (ENTITY+)>
<!ELEMENT ENTITY (VALUE+, SUB_ENTITY*)>
<!ATTLIST ENTITY
name CDATA #REQUIRED
field CDATA #REQUIRED
type CDATA #IMPLIED
>
<!-- the values that this field will be set to for this entity. Many
of these can be specified -->
<!ELEMENT VALUE (#PCDATA)>
<!-- allows nesting of entities - an entity can include many
sub_entities. Many of these
can be specified -->
<!ELEMENT SUB_ENTITY (#PCDATA)>
```

8 Appendix C: et_passwd.dtd

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- DTD for user-based authentication -->

<!ELEMENT ET_PASSWD (USER*)>

<!-- one entry for each user -->
<!-- logons is an optional field containing the number of
concurrent
     logons that this user is allowed to have.
      default will be to allow as many concurrent logons as
possible. -->
<!ELEMENT USER EMPTY>
<!ATTLIST USER
      name        CDATA #REQUIRED
      logons      CDATA #IMPLIED
      pass        CDATA #IMPLIED
      sigkey -id  CDATA #IMPLIED
      sigcheck    CDATA #IMPLIED
      admin       CDATA #IMPLIED
>
```

9 Appendix D: Debug levels and messages

The following tables list the messages that might be received at the various values that the debug-level option can be set to (see page 19).

Square brackets indicate the contents of variables.

9.1 CRITICAL debug level messages

```
CRITICAL: Problem opening libexpat: [text]
```

Table 9-1: CRITICAL debug level messages

9.2 ERROR debug level messages

```
ERROR: [filename]: [text]
ERROR: Cannot open file <[filename]> for reading
ERROR: Deformed group tag for user <[group name]>
ERROR: xmluser refcount ([count]) doesn't match rttpd refcount
([count])
```

Table 9-2: ERROR debug level messages

9.3 NOTIFY debug level messages

```
NOTIFY: Bad XML received - entity received outside user tag
NOTIFY: Bad XML received - MAP received outside USER tag
NOTIFY: Bad XML received - perm received outside user tag
NOTIFY: Bad XML received for user [user name] - entity outside perm
tag
NOTIFY: Entity <[entity name]> not found
NOTIFY: Field [field number] not found for [object name]
```

Table 9-3: NOTIFY debug level messages

9.4 WARN debug level messages

```
WARN: Incorrect number ([number of arguments supplied]) of arguments
given for UDP verbose command
```

Table 9-4: WARN debug level messages

9.5 INFO debug level messages

```
INFO: [object name] passed cross-entity check for entity [entity
name] + [entity name]
INFO: [object name] passed only PRODUCT, not FEELIABLE entity
[entity name]
INFO: [object name] passed simple entity check for entity [entity
name]
INFO: [Allowing/Denying] user <[user name]> access to monitoring
system
INFO: [Allowing/Denying] user <[user name]> access to protected web
pages
INFO: [object name] passed composite <[entity name]> check for user
<[user name]>
INFO: Adding new entry <[entry name]> type [entry type]
INFO: Adding timed event at <[time]> in [number of seconds] seconds
INFO: Attempting to load libexpat from [expat library path]
INFO: Check read for <[user name]> <[object name]> type [object
type]
INFO: Check read permissions for default user (was <[user name]>)
for [object name]
INFO: Check update permissions for default user (was <[user name]>)
for [object name]
INFO: Check write for <[user name]> <[object name]>
INFO: Check write permissions for default user (was <[user name]>)
for [object name]
INFO: Couldn't find matching PROD/SPEC for [object name] (perm
value [permissioning field value]) - matched FEELIABLE entity
[entity name]
INFO: Load of libexpat was successful
INFO: Logging in user <[user name]> as ([default user name]) app:
<[application id]>
INFO: Object <[object name]> is news type, returning OK
```

```
INFO: Object <[object name]> not yet available
INFO: Reading .xml file <[file name]>
INFO: Rereading .xml files
INFO: Rereading .xml files
INFO: Supplied password for user <[user name]> doesn't match
INFO: User [user name] is not permitted to login to monitoring
system
INFO: User [user name] has been deleted
INFO: User <[user name]> has exceeded logon count ([number of
permitted logons]) ejecting [number or sessions] session(s) of
<[application id]>;
INFO: User <[user name]> has exceeded max number of logons ([number
of permitted logons])
INFO: User <[user name]> is [granted/denied] read access to <[object
name]>
INFO: User <[user name]> is [granted/denied] write access to
<[object name]>
INFO: User <[user name]> no longer exists, kicking out [user's file
version]< [current file version]
INFO: User <[user name]> not known, trying default user <[default
user name]>
INFO: User <[user name]> not known
INFO: Mapping <[user name 1]> to <[user name 2]>
```

Table 9-5: INFO debug level messages

9.6 DEBUG debug level messages

```
DEBUG: <[object name]> failed subject check for [object name pattern]
DEBUG: <[object name]> passed subject check for [object name pattern]
DEBUG: Attempting to log out before logging in
DEBUG: Calling invalidate all sessions
DEBUG: Checking group <[group name]> for <[user name]>
DEBUG: Entity <[entity name]> is no longer current, deleting
DEBUG: Entity <[entity name]> is not known
DEBUG: Group [group name] is no longer current, deleting
DEBUG: Object <[object name]> type [object type number]
DEBUG: User <[user name]> has expired ( [user's file version]<
[current file version] ), deleting user
DEBUG: User <[user name]> level [user's file version]< [current file
version] deleting user
DEBUG: User <[user name]> Object <[object name]> Perm = [entity name]
DEBUG: Value for field [field number] for obj [object name] is [field
value] checking entity <[entity name]>
```

Table 9-6: DEBUG debug level messages

10 Appendix E: Example xmlauth.conf

```
##########################################################
#
#  Example Config File for XMLauth
#
##########################################################

##########################################################
#
# Debug level - DEBUG,INFO,WARN,NOTIFY,ERROR,CRIT
#
##########################################################
debug-level INFO

##########################################################
#
# Configuration of where the .XML files are
# (these should include full paths)
#
##########################################################

user-file   /opt/Liberator/etc/users.xml
```

```
passwd-file /opt/Liberator/etc/passwd.xml

entity-file /opt/Liberator/etc/entity.xml


############################################################
#
# If the entity keys are numeric enable this option
#
############################################################
numeric-entity

############################################################
#
# Default user and operation mode
#
############################################################
#default-user+
# Boolean for extending all users permsisions
#default-perm

############################################################
#
# Time to automatically reload the .xml files
#
############################################################
#reload-time02:00:00

# invalidate sessions after reading in .xml files
#eject-after-read
```

Contact Us

Triton Court
14 Finsbury Square
London EC2A 1BR
UK
Telephone: +44 20 7826 9600
Fax: +44 20 7826 9610

www.caplin.com

info@caplin.com