



AMX UNIX / Linux Solutions Guide

Version 2.3

Naviga provides this documentation without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose.

AMPS, AMPS Lan, AMPS Net, AMX, COGENT and ENQUE are trademarks of Naviga.

Other brands and product names used in this manual may be registered trademarks or trademarks of their respective holders.

Table of Contents

Preface.....	1
Introduction	2
Overview of the Solutions Manual	2
Organization of This Guide	2
Overview of UNIX/Linux Server Solutions	4
AmxViewer.....	4
wndpget	4
AMX WNDP C API.....	4
Summary of Solutions	5
AMX Solutions – Solution Format	6
AmxViewer.....	7
General Description	7
Installation.....	7
Use Instructions.....	7
Startup/Shutdown	8
System status	9
Appendix A	10
Appendix B – AMX Hierarchical Directory Structure	11
Appendix C – Story Identifier	13

Preface

Naviga Transmit (AMX) makes it easy to add customized, real-time, rich media content to your website. From thousands of media sources, we bring today's top headlines, and more to your website.

About This Manual

This document describes different ways that the AMX software can be used and integrated with your system. AMPS is a TCP-based protocol used to deliver data over the Internet.

Typographical Conventions

This manual uses the following typographical conventions:

- *install-dir* represents the directory chosen at time of installation.
- **Bold** is used for names the user can enter; for example, all command names and filenames.
- A fixed space font is used for examples. Where user input needs to be distinguished from program output, **bold** is used for user input.
- Nonprinting characters and keyboard characters are capitalized and appear as follows: <EOF>, <CTRL-G>.
- [] Brackets enclose optional items in format and syntax descriptions.
- | A vertical bar separates items in a list of choices.
- # at the beginning of a line signifies a command line prompt.

Customer Support

If you have any problems with the software or documentation, please contact Naviga Customer Support via telephone or email as described below.

Support Location	Telephone	Email
Roseland, NJ, USA	+1 (973) 422-0800	datasupport@navigaglobal.com

Introduction

Overview of the Solutions Manual

This is the UNIX/Linux Server Solutions Guide for Naviga AMX. This guide is designed to answer the question “I’ve successfully installed AMX. What are my next steps?” It describes different ways that the AMX software can be used and integrated with your system. You may choose to use one, several or none of the solutions in this manual.

Note: This document discusses various ways to make use of the news and information provided by Naviga AMX. Clearly, these are not the only solutions possible.

Organization of This Guide

You will need a basic understanding of the AMX system. This is provided in the *Overview of the Naviga AMX for UNIX/Linux* section. Also, refer to documents in the *Related Documentation* section for further information on AMX.

The *Overview of UNIX/Linux Server Solutions* section provides a brief overview of the various solutions discussed within this guide. You should read through this section and choose the solutions you would like to understand in more detail. You can then go to the major section on the chosen solutions and read more detailed information about how to implement the solution at your location.

Overview of Naviga Transmit (AMX) for UNIX/Linux

The AMX software consists of:

- **AMX Interceiver.** This is a daemon (named `intcvr`) that maintains a connection between your system and the AMPS server at the Naviga data center location in Newark, NJ. This connection is used to transmit news and other information to your system.
- **WNDP Server.** This daemon (named `wndp`) caches the data (news) files created by the AMX Interceiver. It then provides a generic interface whereby one or more external programs (usually other daemons) can receive and process the news using the WNDP protocol.
- One or more applications that process the XMLNews stories and associated metadata files (RDF files).

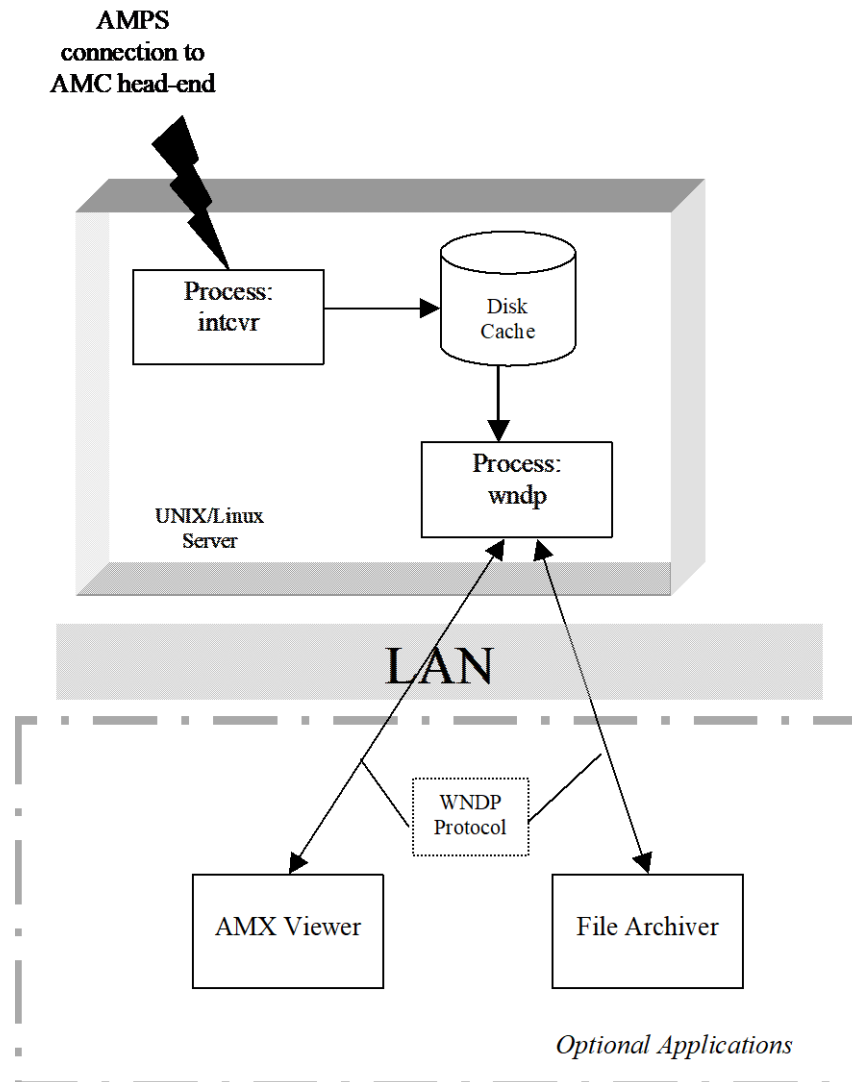


Figure 1 Overview of AMX Components

As shown in Figure 1, the AMX WNDP Server accesses the disk cache of news and information that was received over the AMPS connection between the AMX Interceiver and the Naviga AMPS server. One or more applications may then receive the information from the AMX WNDP Server using the WNDP Protocol. These applications will usually be other daemons but may be any type of application that can establish a TCP connection to the WNDP Server. You may write these applications, or you may choose to use applications that are provided by Naviga (as described in this document). The File Archiver is an example of an application that connects to the AMX WNDP Server. It is described in more detail in this document. The news is transmitted to the attached applications using the Naviga implementation of the NITF XML format, plus additional metadata in a RDF format.

Overview of UNIX/Linux Server Solutions

AmxViewer

AmxViewer is a utility program (amxviewer) that can be used by the local AMX server administrator to monitor the AMX Service. It provides a basic data scroll that can be used to indicate that content is currently being received; it is a simple terminal application only. Use amxviewer if you need the ability to monitor the news coming through the AMX service. It is not an end-user solution. NOTE: all non XMLNEWS-STORY class content will *not* display the headline. The time, class, filename and size in bytes will be displayed instead.

wndpget

The wndpget process provides the ability to save news into a directory or hierarchy of directories. Look at /nwsys/release/conf/wndpget.conf for additional configuration options.

AMX WNDP C API

The AMX WNDP C API provides a library of routines to simplify the task of building custom applications using the Naviga AMX WNDP Protocol. If you are writing applications in C, you can use the AMX WNDP C API to simplify the task of connecting to the AMX service and receiving the XML news.

Use the AMX WNDP C API if you want to write a custom C application to process XMLNews stories directly. You will need to be an experienced C developer to use this solution. This API is the recommended interface for using WNDP; we do not suggest writing directly to a socket.

Summary of Solutions

The following matrix summarizes information for the above solutions:

Program	General Description	Degree of Solution	Also requires:	Installation file
AmxViewer	Display incoming news	Administration only		amxviewer
AMX WNDP C API	Provides programming API to connect to the WNPD Server and collect stories	Requires additional C development to use		libclient.a libcomm.a libtext.a wndpcomm.h

AMX Solutions – Solution Format

Each solution is described with the following headings:

1. General Description
2. Installation
3. Use instructions

Note: Do not install any of the solutions outlined in this guide until you have completed installation of the required components of the AMX software (see the AMX UNIX/Linux Installation Guide).

AmxViewer

General Description

AmxViewer is a utility for displaying real-time information as it is being received. It is intended to be used as part of an administrator's console to verify that news is flowing over the Internet from the Naviga AMPS head-end to the server on which the AMX Interceiver is installed. AmxViewer installation is not required for the rest of the AMX software to operate correctly. However, it is useful in ensuring that the system is functioning properly.

Installation

To install the AmxViewer software, follow the instructions in the AMX UNIX/Linux Installation Guide.

Use Instructions

1. Start up AmxViewer by running the executable amxviewer (found in the directory chosen at the time of installation.) *Note: if the mxp.profile startup script is being used, amxviewer can be referenced directly without the full pathname. Please refer to the installation guide.*
2. You should see story headlines on your screen. Note that stories are received in real-time. Therefore, if you are running this installation late-at-night, it is likely that it could take several minutes or more before stories appear on the screen. Even during business hours, there may be long pauses between the receipt of stories.
3. Along with each headline, the publication time of the story and the name of the news provider are displayed.

Example XMLNEWS-STORY reception:

```
19:01 MKTNEWS    UK Data Snap: Private Sector Pay Creeps Up In Nov - CBI
      20031221̄ (receivedTime: 2003-12-24T13:10:08-05:00)
13:12 MKTNEWS    BULLET: US TSYS/RECAP: Market crept higher amid a host of...
      20031224 (receivedTime: 2003-12-24T13:12:47-05:00)
13:21 MKTNEWS    BULLET: Tsy Futures: Maturities end better, reversing most..
      20031224 (receivedTime: 2003-12-24T13:21:54-05:00)
```

NOTE: The date below the news provider is the story publication day. This is derived from the <xn:publicationTime> tag in the RDF file. This may be different from the receivedTime field, which is updated during actual receipt of data on your server. In the above example, the first line shows a retransmission of an older story.

Example non-XMLNEWS-STORY reception:

```
17:15 HTML       File: 200312191715MKTNEWS_MAINWIRE_B1BE_2651.html (3657 bytes)
23:57 PLAIN-TEXT  File: 200312212357MKTNEWS_BULLETXI_A2E5_2656.txt (642 bytes)
```

NOTE: non-XMLNEWS-STORY content is displayed via the actual filename and size in bytes of the corresponding disk file.

Startup/Shutdown

When optional components have been installed, the `amxopt` utility is required for start, stop and status. This utility resides in the directory `/var/adm/amx/bin`. It is recommended that this directory be added to your *path* environment variable. If processes are already running, the start command will have no effect on the system.

For the following examples, the optional components `filearch`, `indexer` and `scroller` have been installed. It is also important to note that these three components are the only ones that require `amxopt` for starting and stopping.

```
{nadmin} > amxopt
```

```
Usage: amxopt [info|status|start|stop [-force]]
```

```
info           - Summarizes which products are installed.
status         - Displays process status.
start          - Starts all installed optional products.
stop [-force]  - Stops all installed optional products.
                  Use -force to avoid prompt.
```

```
{nadmin} > amxopt start
```

```
<<< Starting File Archiver (1.1) >>>
Starting /pool/amx/programs/xsltd
Starting /pool/amx/programs/filearch
```

Use `amxopt stop` to halt all running optional components that have been installed. Use the switch `-force` to avoid being asked the yes/no question for each component.

```
{nadmin} > amxopt stop
```

```
Stop 'File Archiver '? (y/n) y
```

```
<<< Stopping File Archiver (1.1) >>>
Stopping process filearch ... done
Stopping process xsltd ... done
```

Note: Each of the two processes that make up the File Archiver (`xsltd` and `filearch`) forks a child process to handle each incoming data file. (Each child process will handle exactly one data file.) The child processes are configured so that they will not respond to `'amxopt stop'`. `'amxopt stop'` will stop the parent `filearch` and `xsltd` processes so that no additional children will be forked and each of the currently running child process will exit when it finishes processing its data file. The child processes usually live for only seconds but are guaranteed to live no longer than 1 minute.

System status

Use the amxopt utility with the info and status switches to view current state of the system:

```
{nadmin} > amxopt info
```

```
Current installation information Wed May 24 16:06:47 EDT  
2003
```

```
Application  :    AMX  
Release     :    1.1  
User acct:   nadmin  
Location    :    /pool/amx
```

```
Application  :    File Archiver  
Release     :    1.1  
User acct:   nadmin  
Location    :    /pool/amx
```

```
{nadmin} > amxopt status
```

```
<<< Application: File Archiver  (1.1) >>>  
xsltd:                UP  
filearch:              UP
```

Appendix A

This appendix is intentionally left blank for compatibility with other versions of this manual.

Appendix B – AMX Hierarchical Directory Structure

AMX optionally archives XML-encoded stories and/or HTML-encoded stories in a hierarchical directory structure. We have attempted to maintain a common structure across all our platforms. This appendix describes that structure.

For all the applications, you can specify a single installation directory. This directory is designated as “*install-dir*” below. Directories are created as needed by each application. Therefore, not all directories are created by all applications.

Note: In the discussion that follows directory names which appear in braces (i.e., “{” and “}”) are references to names which are typically configurable by you or do not have a fixed format. Directory names that are not in braces are the subdirectory names that will literally appear within the directory structure.

The directory structure, which is described in more detail below, appears as follows:
{*install-dir*}

```
|__programs
|   |__cfg
|   |__utils
|   |__logs
|   |__templates
|__web
|   |__data
|   |   |__html
|   |   |   |__{YYYYMMDD}
|   |   |   |   |__{HH}
|   |   |__xml
|   |   |   |__{YYYYMMDD}
|   |   |   |   |__{HH}
|   |   |__bad
|   |   |   |__{YYYYMMDD}
|   |__cgi
|   |__images
|__incoming
|__tmp
```

The first level of directories below the {*install-dir*} directory is described below.

programs – This directory contains any executables being used by the server application. This directory will contain a subdirectory called *cfg* that contains configuration files and a subdirectory called *templates* that contains the stylesheets

files used to convert the XML and RDF files into html files. It may also contain a subdirectory called “utils” which contains utility programs and a subdirectory called “logs” which contains log files generated by the Naviga server applications.

programs/logs - This directory will contain up to seven files (sun mon tue wed thu fri sat) which are named based on the day of the week. This logging scheme was chosen so that a full week of historical logs would always exist. Old logs are automatically truncated when the first message of a new day is written.

web – This directory contains information which may be published to users. As described in more detail below, the XML and HTML files appear in subdirectories below this directory.

web/cgi – This directory contains cgi programs used by the Index Builder Web Site application.

web/images – This directory contains image files used by the Index Builder Web Site application.

incoming – This directory is used by the AMX Index Builder. See a description of this service for more information.

tmp – This directory is used for temporary files created by the various applications.

The “web” subdirectory contains several subdirectories that are used to archive XML and/or HTML versions of news stories being received via AMX. There are also other directories that are used if you are using the AMX Index Builder. The directories that appear under the “web” subdirectory are as follows:

data – This subdirectory contains a hierarchy of directories used to store the XML and/or HTML versions of the stories. If there are XML versions of the stories, they are contained in the “xml” subdirectory within the “data” subdirectory. If there are HTML versions of the stories, they are contained in the “html” subdirectory within the “data” subdirectory. The subdirectories under these directories are based on the publish date/time of the story (normalized to Eastern Time). There are two levels of subdirectories under the HTML and the XML subdirectories. The first level is based on the publication date. The form of the directory name is “{YYYYMMDD}” where YYYY is the 4-digit year, MM is the 2-digit month, and DD is the 2-digit day. For example, an XML story with a publication date of August 7, 1999 would appear in a subdirectory with a name “19990807”. This subdirectory would have a second level of directories based on the publication hour. For example, a story published at 4:27 p.m. (Eastern time) would appear in a subdirectory called “16”.

Appendix C – Story Identifier

Each XML story is tagged with a story identifier. This story identifier is used as the filename. The XMLNews story filename is the story identifier followed by the extension “.xml”. The XMLMeta file name is the same story identifier followed by the extension “.rdf”.

The format of the story identifier is as follows:

yyyymmddhhnnppppppppppssssssssuuuuuuuuuuuuuuuu

This format is described in detail below.

- A date/time based on the publication date/time. The format for this is:
yyyymmddhhnn
Where yyyy is the 4-digit year, mm is the 2-digit month (January = 01, February = 02, and so on), dd is the 2-digit day (e.g., 01, 02, ..., 10, 11, ...), hh is the hours (represented in 24-hour military time), and nn is the minutes. The date/time is in Eastern Time and reflects the publication time of the story.
- pppppppp is the 8-character information provider string. These strings map directly specific providers. So, a provider will always have the same provider string.
- ssssssss is the 8-character information provider service string. These strings map directly to information provider designated services within the information provider’s feed.
- uuuuuuuuuuuuu is a unique string added to the story identifier to ensure uniqueness of the identifier among stories sent by the same information provider in the same service at the same time. In some cases, the information provider provides the unique string. In other cases, the identifier is generated automatically by the system. Often, though not always, this will be a 13-character string.

To summarize:

- characters 1 through 12 are always the date
- characters 13 through 20 are always the provider designation
- characters 21 through 28 are always the service designation
- the remaining characters are a unique string to ensure uniqueness of the identifier